3-2022

System Phasing and Schedule Growth Analysis

Daniel A. Long

SOFTWARE PHASING AND SCHEDULE GROWTH ANALYSIS

THESIS

Daniel A. Long, Captain, USAF

AFIT-ENV-MS-22-M-228

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENV-MS-22-M-228

SOFTWARE PHASING AND SCHEDULE GROWTH ANALYSIS

THESIS

Presented to the Faculty

Department of Systems Engineering & Management

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Cost Analysis

Daniel A. Long, BSME

Captain, USAF

24 March 2022

AFIT-ENV-MS-22-M-228

SOFTWARE PHASING AND SCHEDULE GROWTH ANALYSIS

Daniel A. Long, BSME
Captain, USAF

Committee Membership:

Lt Col. Scott Drylie, PhD
Chair

Dr. Jonathan Ritschel, PhD
Member

Lt Col. Clay Koschnick, PhD
Member

Abstract

Software development research, once a priority for the DoD, has received less focus in recent years. What research that has occurred has focused on size and cost prediction of software. Generally lacking in these studies is analysis on phase distributions and schedule. Putnam (1978) showed that there were measurable effects between early program management and final schedule growth, but these relationships have not been explored using the 2001-2021 DoD Software Resources Data Report (SRDR) database. Additionally, industry software development guidance provides rules of thumb for effort allocation, but a comparison of the rules to DoD software projects is nonexistent. This thesis seeks to understand how the development phases of software interact with each other and with final outcomes of schedule and effort, as well as provide guidance for the program manager and developer.

Using least square models and Hotelling's T^2 test we evaluated conventional rules of thumb for effort allocation against projects in the SRDR database. Given the variation in the data, we find that multiple rules of thumb are applicable to DoD software development. We then compared how the effort allocation varies between projects with either high or low schedule growth. Our analysis shows that increasing effort in early phases decreases the total schedule growth. These finding were significant across multiple categories such as developmental process type, Service, and project size.

Acknowledgments

I first want to thank my thesis advisor, Lt Col Drylie, for his persistent editing and advice. No sentence was left unturned after months of editing, and this thesis draft is all the better for it. I hope the stress of last-minute testing was not enough to convert you from a vegan diet. I would also like to thank my committee members, Dr. Ritschel and Lt Col Koschnick for their input and time throughout the process.

To my wife, thank you. You helped out in more ways than you know, taking care of the family while I struggled with statistical anomalies. And to my daughter, you will always be the best thing that happened here at AFIT in the middle of a pandemic. Your morning smiles are all I need.

Daniel A. Long

Table of Contents

Page

List of Figures

List of Tables

SOFTWARE PHASING AND SCHEDULE GROWTH ANALYSIS

I. Introduction

1.1 Background

The Department of Defense (DoD) is no stranger to problems revolving around software development. According to a recent report, 20 of 29 reported DoD software programs experienced cost increases and/or schedule delays since Jan 2019 (GAO, 2021). The report cited factors such as modernization, requirement changes, technical complexity, and contracting issues. But, the problems likely run much deeper. In an indictment of the entire acquisition process, the Air Force's first Chief Software Officer abruptly resigned in 2021, complaining of "borderline criminal" practices (Roza, 2021). While there has been regular effort to understand the factors of success, new perspectives on how to manage software are clearly needed.

There have been some empirical studies on DoD software, primarily by Air Force Institute of Technology (AFIT) and Naval Postgraduate School students. The earlier work in the 1980's and 90's focused on estimating software size and evaluating the effectiveness of proprietary software estimating packages(Blalock, 1988; Coggins, 1993; Daly, 1990). Recently, studies have examined factors contributing to software cost growth (Violette, 2021) and project efficiency (Amato, 2021). None have looked at the relationship of the various phases to better understand the underlying mechanics of their effect on final outcomes of schedule or cost.

Two perspectives on software may aid the DoD. First, a sense of the amount of effort DoD might expect to have to plan for each phase of a project. The software

development process typically has distinct phases, ranging from initial requirements definition to testing and evaluation. Industry papers, guidelines, software textbooks, and multiple research papers cite various Rules of Thumb (henceforth, RoT) for how much effort to allocate to each phase (Ambler, 2005; Boehm, 1987; Brooks, 1975; Pressman, 2001; Sommerville, 2016; Thibodeau and Dodson, 1980; Zelkowitz, 1978). Generally, there is a lack of empirical support for these effort allocations. A commonly cited RoT in the literature is a 20/20/30/20/10 split of effort among the phases for requirements/analysis/coding/testing/transition. The first identifiable source for this split, however, is a blog post in 2005 and provides no support other than "industry experience" (Borysowich, 2005). Another common RoT is a 40/20/40 split among analysis & design/coding/testing & integration, a rule recommended by the DoD's software estimation manual. As yet, we cannot find a source for this division nor the data to back these two RoT up.

A number of researchers have attempted to validate these rules and propose new ones (Heijstek and Chaudron, 2008; Jolak et al., 2018; Papatheocharous et al., 2017; Yang et al., 2008), though the data used are often small in scope or represent a concentrated study area. Of the empirical works, none have a sample size greater than 100. Particularly problematic, none of the RoT comparisons use government defense data. RoT have long had a role in program planning and management, but practitioners should have some confidence that the rules are founded in real data and relevant to the DoD. The first aim of this paper is to attempt to validate the applicability of these RoT to Defense programs, and propose a new rule if appropriate.

The second perspective on schedule which may aid DoD is an understanding of which features of schedule relate to success and failure. Again, there is literature to consider. Putnam (1979) applied the Norden-Rayleigh Curve (commonly referred to as the Rayleigh curve) to the distribution of effort for software, predicting particular

patterns in development. This curve maps the rise and fall of man-hours allocated to a project over time and has been used in Putnam's proprietary software (SLIM) to estimate costs and schedules for new starts. He showed that there were measurable effects between early program management and final growth. Thibodeau (1980) documented similar relationships between effort and cost for DoD software programs. In these studies, schedule is an integral factor in active program management. More effort in early development phases may improve the project's chances of success. Brooks (1975) found that compressing schedule or increasing effort to alleviate development issues may compound schedule and cost problems. Software and its uses have metamorphosed radically in the last three decades. Relevant empirical support sustaining Putnam's conclusions is difficult to find for contemporary projects and methods. The relationship between phases and outcomes has not been fully explored in modern DoD software projects and will be the focus of the paper.

Within the DoD, some work has been done on scheduling major aircraft acquisition programs, but its applicability to software is limited (G. Brown, 2015; T. Brown et al., 2002; D'Amico, 2017; Unger, 2001). Regarding software, work has tended to focus on overall acquisition processes like comparing developmental methods (e.g. Agile and Waterfall) rather than exploring the mechanics of phasing and schedule (Dayal Chauhan et al., 2018; Goljan, 2021; Prasetya et al., 2021; The Standish Group, 2015). Therefore, the second aim of this paper is to elucidate the relationship between phases and schedule.

The graph in Figure 1.1, for a single development case, serves as an example of the hours of effort spent on each phase of development. It also exemplifies some of the error that occurs in scheduling. The two lines represent the initial estimate (blue) and the final report (red). The first two phases—focused on overarching design-have a relatively low amount of effort, then the coding ramps up in the third Coding & Testing

phase. Coding effort then tapers down as the bulk of the code has been written and undergoes integration, quality control, and evaluation. In both the initial estimate and final report, the pattern is the same, which is a promising indication that project managers have a general sense of the amount of effort for each phase. And interestingly, the curve bears some superficial resemblance to the Rayleigh curve. That being said, the difference between the curves shows that each phase was underestimated, with the largest delta in phase 3.

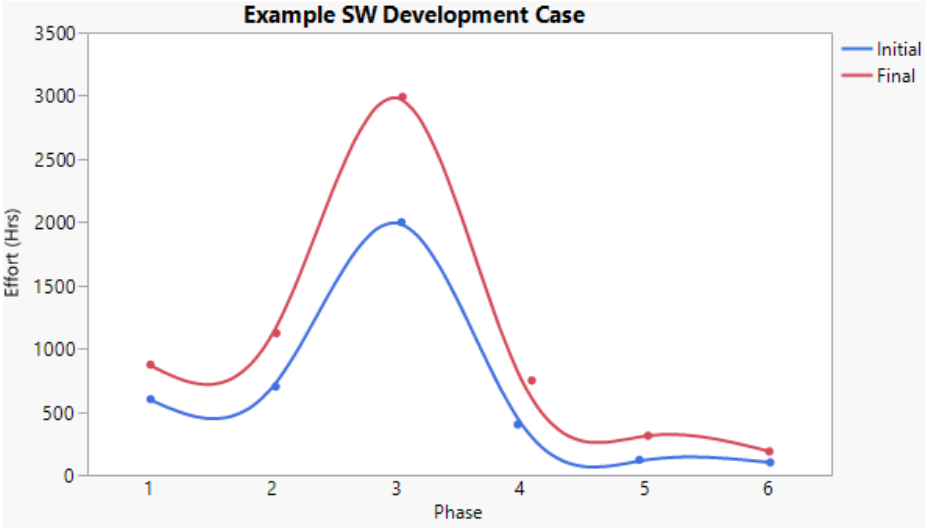


Figure 1.1: Example distribution of project effort

1.2 Project Scope

The following questions will be investigated in this study:

1. What phase characteristics do DoD software projects show? How do they compare to general Rules of Thumb?

2. How accurate are we with phase planning? (Do we systematically get our predictions of phasing wrong compared to actuals?)

3. How does effort allocation differ between low schedule growth and high schedule growth projects?

4. How do errors in schedule effort in early phases relate to errors in subsequent phases?

5. For the above questions, are there statistically different results for different categorical parameters (Waterfall vs. Agile, Service, Domain, etc.)?

1.3 Methodology

The most recent version of the Software Resources Data Report (SRDR) is used as the primary data source. The database covers 21 years of software development and requires some cleanup to isolate quality projects with all data values tabulated. After cleanup, initial analysis is conducted by plotting the relationships between expected and final hours & time for each development stage. Existing RoT are compared to the data to identify how well the initial estimates and final outcomes match the RoT breakouts, if at all. Any patterns found are validated through Least Means Squared and Hotelling's T^2 test. Next, schedule growth is calculated for each project and averaged. We then compared high- and low-growth cohorts to determine if early phasing efforts have an effect on growth. Further investigation for all methods is conducted on various categories within the database, such as Service, Application Domain, ESLOC size (large vs small), etc.

II. Literature Review

> *The current approach to software development is broken and is a leading source of risk to DoD: it takes too long, is too expensive, and exposes warfighters to unacceptable risk by delaying their access to tools they need to ensure mission success.*
> *- Defense Innovation Board, 2019*

The Standish Group has maintained one of the largest available databases of software development performance since the 1990s (The Standish Group, 2015). Their findings over the past two decades indicate the average project success rate is only 33 percent. About 19 percent of projects are considered failures, and the remaining 45 percent of projects have cost and schedule overruns or produce low-quality results. When broken out by sector, depicted in Figure 2.1, the reports show the worst-performing sector is Government with the highest failure rate at 24 percent and the lowest success rate at 21 percent (The Standish Group, 2015). The difficulty of software development exists in both private and public sectors, but the government is still the worst-performing sector across the board. The figure also reflects the Government's own report on software projects, which found that two-thirds of new DoD software starts were behind schedule or over cost (GAO, 2021). Therefore, it is necessary to study successful programs to determine ways in which software estimation can improve schedule and cost results.

	Successful	Challenged	Failed
Retail	35%	49%	16%
Banking	30%	55%	15%
Financial	29%	56%	15%
Healthcare	29%	53%	18%
Services	29%	52%	19%
Other	29%	48%	23%
Manufacturing	28%	53%	19%
Telecom	24%	53%	23%
Government	21%	55%	24%

Figure 2.1: Sector Performance, 2011-2015 (Standish 2015)

2.1 Software Cost and Schedule

The extant empirical research on estimating software total cost and time is relatively robust. In the 1970's Norden and Putnam approximated the distribution of software development over the course of a project's life (Putnam, 1978). The man-hours assigned to a project began slowly, peaked around 40 percent of the way complete, and dwindled as the project entered final testing and quality assurance. The resulting distribution fit a Rayleigh Curve (seen in Figure 2.2) and has been utilized in software estimation and program management for decades. In the graph, Manpower (effort) is measured on the vertical axis and periods of time are on the horizontal.

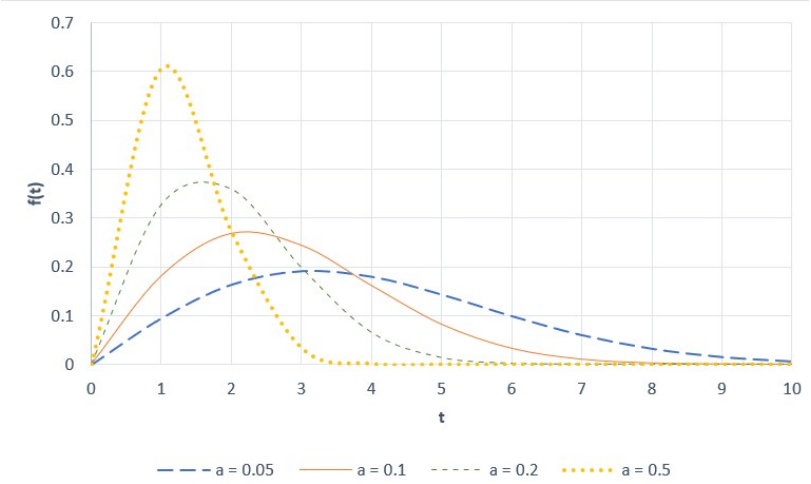


Figure 2.2: Rayleigh Curve with Varied Scale (a) Parameters

The Rayleigh curve is defined by equation 2.1:

$$f(t) = 2ate^{-at^2} \tag{2.1}$$

where

a = Rayleigh scale parameter

t = time elapsed

The scale parameter a adjusts the curve to reflect where the peak effort is in the project. As depicted, higher initial peaks lead to projects which finish earlier than on projects where the peaks take longer to form. Choosing the right curve, or distribution of effort, is important for program managers and bureaucrats alike. For one, software development delays can have impacts beyond the scope of the software itself, such as delays in the full hardware system it will be integrated with. Two, delays increase the manpower hours and thus increase the labor costs for a project (AFCAA; NCCA, 2008; Boehm and Basili, 2001; Brooks, 1975; Sommerville, 2016). Lastly, the manpower assigned cannot be used elsewhere, potentially resulting in lost opportunity for the organization. In this manner, schedule increases are positively

correlated to cost increases. Depending on the contract type and frame of reference, the relationship can be considered a causal one.

Putnam further relates effort to schedule through Equation 2.2, where effort is a function of size, a B skills factor, productivity factors, and time (Putnam, 1991). The **B** factor is derived from Putnam's research and is based on skills and SLOC; a larger SLOC incorporates a larger B value, anywhere from 0.1 to 0.5. Similarly, the productivity parameter is based on application types (e.g. radar systems v. telecommunications) and exponentially increases from 1 to over 300k. Larger parameter values are associated with better-performing teams and program management, thus reducing total effort.

$$\textit{Effort} = \left[\frac{Size \times B^{1/3}}{Productivity\ Parameter}\right]^3 \cdot \frac{1}{Time^4} \tag{2.2}$$

The effort equation can be further simplified. Assuming the productivity of the team is constant, and since B is a factor of SLOC, the first portion can be reduced to a constant times size. Ignoring exponents and constants, the general relationship can be described as a product of size over time. This is shown in Equation 2.3. As size increases, so does total effort. As time increases, effort decreases, ceteris paribus.

$$\textit{Effort} = \frac{Size}{Time} \tag{2.3}$$

2.2 Methods of development

Implicit in Norden and Putnam's work is that there are different phases of activities, requiring different levels of effort, and that early decisions can impact later phases. The phases are typically designated as Requirement Analysis, Design, Coding, Testing, and Transition. Comparably, and focused on in this thesis, the phase distribution used by the DoD is listed below:

1. Requirements Analysis
2. Architecture & Design
3. Coding & Testing
4. System Integration
5. Qualification Testing
6. Development Test & Evaluation
7. Other Development Effort [1]

Though the activities involved in the seven DoD phases are not formally defined in any government handbook, the definitions can be gleaned from excerpts and outside literature like Zelkowitz (1978), Yifftachel et al. (2011) and textbooks from Fairley (2008) and Sommerville (2016). In the Requirements Analysis phase, the team identifies all activities and results the software should accomplish, as well as the expected resources required in the process. In the Architecture and Design phase, the team outlines the end structure of the code, connecting nodes between code modules/components, and the methods of obtaining the intended results. The bulk of development, at least in terms of lines of code written, is typically performed in the Coding and Testing phase. The team codes the program itself that will obtain the desired results for the customer, then tests the code for functionality.

System Integration verifies performance against the requirements in the final environment of the software. If the software will interact with multiple external systems, the primary validation of a successful handshake between systems is completed in this phase. As described by Pressman, the Qualification Testing phase evaluates the

[1]Phase includes Configuration & Program Management, Quality Assurance, Process Implementation, & other Support processes; not utilized in this thesis

software through the lens of Verification and Validation (Pressman, 2001). Verification asks if the right software is being constructed (will it do what we want?), while Validation asks if software is being developed in the right manner (i.e. Will the customer be happy?). Finally, Development Test & Evaluation is performed to test for bugs and errors in functionality before the software is handed off to the customer and their operations and maintenance team (Sommerville, 2016).

There are two primary ways of completing the tasks in these phases: Traditional and Agile methods. In the Traditional (a.k.a. "Waterfall") development process, the team collectively moves the project sequentially from phase to phase (Pressman, 2001). The benefit of the Traditional model is that the linear layout makes it simple to recognize where the project is at in development at any one time. It also forces the developers to attempt to fully flesh out each phase before continuing, since there is usually no backtracking of earlier phases. This facet is simultaneously one of the model's main drawbacks; by forcing each phase to be fully completed before moving to the next, it encourages over-optimization and discourages quick prototyping and testing (Putnam and Myers, 1992). In this sense, Traditional development may be the best model for a safety system that cannot be allowed to fail, but may not be the best for video game design or web app development.

The Agile model was designed to ameliorate the unforgiving rigidity of the Traditional model. Agile developers accomplish smaller packages of product for testing and delivery, allowing quick feedback loops of quality and the happiness of the customer (Sommerville, 2016). It encourages rapid movement forward and backward between the phases. If a project sees multiple gaps in the system integration phase, the process is quite conducive to backtracking and fleshing out the architecture and coding of the project. Typically, the agile model is effective at producing a limited, workable software solution for fielding while simultaneously improving on and building up the

software. After a slew of minor iterations, the software is built to standard but with the benefit of continuous testing and feedback. While the Agile model is effective to getting a minimally-viable product in front of customers/stakeholders for feedback, Traditional may be a better model for designing critical security and safety systems that require certification before use (Sommerville, 2016).

With the modernization of software inevitably comes the modernization of the development process. In the past decade there has been a shift towards cloud computing and always-accessible software updates. Very rarely is a new piece of software developed without consideration for long-term sustainment and improvement. One example is the Microsoft Office suite of applications. In the past, Microsoft developed a stand-alone software package every few years, showcasing the latest updates and graphics. Successive releases introduced new features, but each new package was developed and sold separately from its predecessors. Recently Microsoft moved to a continuous update model (sometimes called "Flow") with M365. The company now develops and integrates features into the existing Office structure, continually updating the software without separate feature roll-outs or disruption to the user.

2.3 Phase Distribution

With the phases and methods so defined, the question then focuses on what is the appropriate level of effort to plan for each phase. Several researchers have offered recommendations, which to varying degrees have been discussed as heuristics or rules of thumbs (RoT). The rules are not intended to be prescriptive but rather rough guides, especially early on. Perhaps the first attempt to outlay effort allotted to each phase was in 1956 by Benington. Based on his experience and research at Lincoln Laboratories, he allotted roughly 50 percent to Program and Coding Specifications, 10 percent to the coding itself, and 40 percent to Parameter and Assembly Testing

(Benington, 1983). The emphasis on the initial phases has some intuitive appeal. Spending more time on properly designing the project than on actually writing the code seems a reasonable proposition, analogous to the benefit of fully drafting a building's dimensions and specifications before cobbling lumber together.

While the origin of this rule has been lost, the earliest found citation of this rule was by Thibodeau in 1980, who himself regards the rule as "often cited" (Thibodeau and Dodson, 1980). Boehm discards this RoT as early as 1987, at least in the context of software development in an Industrial Engineering setting. He instead proposes to use a 60/15/25 distribution "to treat product development as more than just programming" (Boehm, 1987). Given the subsequent decades of research and guidelines, it does not seem to be that many people cited or were aware of Boehm's proposal. In later papers, Boehm uses different rules of thumb but does not appear to mention his first distribution (Boehm, 2000; Tan et al., 2011).

The next source for a RoT is from 1975 by Brooks, again from *The Mythical Man-Month*. Though the book is focused on management and scheduling, he writes that his personal RoT for development is 33 percent Planning, 17 percent Coding, 25 percent Component Test, and 25 percent Systems Test. This rule effectively shifts a portion of Benington's effort from the early design phases to the late testing phases.

Shortly thereafter, Zelkowitz published an overview of current software development processes and principles (Zelkowitz, 1978). In the paper, he put forth a rule of 10 percent each for Requirements and Specification, 15 percent for Design, 20 percent for Coding, 25 percent for Module Test and 20 percent for Integration Test. This broke out the effort among more phases than previous rules of thumb, but closely aligned to the Brooks RoT.

The 2000's saw two new RoT crop up. In the context of IBM's Rational Unified Process for software development, Ambler allocates 10 percent towards Inception,

25 percent to Elaboration, 55 percent to Construction, and 10 percent to Transition (Ambler, 2005). Titled differently, the definitions are still comparable to earlier RoT. Inception and Elaboration map to the requirement planning and design phases, Construction maps to the main Coding phase, and Transition encompasses the same events as integration and test. This rule diverges significantly from previous rules in how much more effort it allots for the middle Coding phase. Previous rules recommended between 15 and 20 percent effort, but Ambler chooses 55 percent.

Striking a middle ground for coding, Borysowich (2005) distributes the effort across five phases. His rule is to apply "15 to 20 percent toward requirements, 15 to 20 percent toward analysis and design, 25 to 30 percent toward construction (coding and unit testing), 15 to 20 percent toward system-level testing and integration, and 5 to 10 percent toward transition" (Borysowich, 2005). This phasing profile is cited in numerous papers as the gospel for software development (Tan, 2012; Tan et al., 2011; Violette, 2021). However, the rule is poorly constructed overall. For one, the stated "ranges" for each phase are not intelligible as ranges. One would have to use the maximum target for each phase in order to sum up to 100 percent of the effort. Using any but the maximum would lead to a summation of less, and as low as 75 percent. Borysowich also characterizes these factors on a personal blog in 2005, as "general industry experience." It is concerning that so many practitioners and researchers have relied on a RoT with dubious origins.

The major textbooks on software development advocate for similar rules of thumb. Pressman's 2001 publication Software Engineering: A Practitioner's Approach recommends the 40/20/40 rule from Thibodeau's era. He does go into more detail by discussing where modifications to the rule should be made depending on what the project entails. In a related vein, Sommerville recommends a 15/25/20/40 split in the 10th edition of Software Engineering, a separate textbook series (2018). This is

the same as the 40/20/40 rule, but splits the first 40 percent effort into 15 percent Specification and 25 percent Design. The DoD's own textbook, Software Development Cost Estimating Guidebook, recommends the 40/20/40 rule without evidence (AFCAA; NCCA, 2008).

While the above rules of thumb are generally considered "expert opinion," there have been recent attempts to empirically derive a rule of thumb from existing data. Heijstek & Chaudron studied the phase profile for 20 industrial engineering software projects in 2008. The mean results allotted 21 percent to Planning & Requirements, 11 percent to Analysis & Design, 38 percent to Coding, 12 percent to Testing, and 4 percent to Configuration. They were surprised to find the effort was so small for the second phase, the Analysis and Design phase, compared to industry guidelines based on Boehm and Ambler's rules of thumb. Unfortunately, they could not offer a significant reason as to why the second phase was so much smaller than expected.

Yang et al. studied the phase distribution of 75 development projects within the Chinese Software Benchmarking Standards Group (CSBSG) database (Yang et al., 2008). They found a distribution of 16 percent Planning & Requirements, 15 percent Design, 40 percent Coding, 22 percent Testing, and 7 percent Transition. The standard deviation for the means was approximately half the mean percentage value. Papatheocharous arrived at nearly the same phase pattern using data from the International Software Benchmarking Standards Group (ISBSG), allocating the respective percentages of 19/14/42/18/7 percent for each phase (Papatheocharous et al., 2017). Papatheocharous had high variation in his dataset, where the standard deviation for each phase was 20% to 300% higher than the mean phase percentage. It is interesting to see that analysis of two geographically separated databases produced such similar results. Full summary statistics for Yang and Papatheocharous' research is in the appendix.

Jolak et al. compiled multiple rules to compare within the framework of Model-Based Engineering (2018). The MBE process emphasizes virtual models and simulations to assist in developing software for physical end items. Jolak had two teams develop software over a period of months and tracked their effort at each stage. The study then compared mean effort in each phase against the aforementioned RoTs. The results were a close match to the 40/20/40 rule, with the students allocating effort via a 44/23/33 percent split.

As for research within the DoD context, Tan et. al. attempted to test a RoT against an earlier version of the SRDR database (Tan et al., 2011). Tan evaluated 257 SRDR projects (as of 2005) specifically against the COCOMO II model distribution. Unfortunately, there is little we can substantively derive from this study. The research used a much earlier dataset from the SRDR, with far less data. Moreover, while they intended to determine if these phase efforts differed by "application type," they do not run the statistical tests necessary to be confident of that factor. Their recommendation to treat application types differently is, despite their recommendation, not statistically supported. This is the only thesis or journal article found to evaluate the SRDR phases in any manner.

The summary impression is one of a literature which does not converge on a singular pattern. Researchers differ substantively in terms of phase schemes (number of phases) as well as proportional assignments. In fact, given that none of the stated RoT even directly mirror the scheme employed in the SRDR, it is not immediately clear that any RoT is substantially relevant to the DoD. The SRDR database, as described earlier, is designed to account for six phase designations. Ambler, however, uses only four, having combined phases 4-6 (Ambler, 2005). Pressman uses just three, combining phases 1-2 and 4-6 (Pressman, 2001). In total, none of the eleven studied RoTs split the testing phase into Qualification Test and Development Test as the

SRDR does. Thus, in order to determine if any rules have value to DoD, the varied schemes must first be assessed for their ability to be translated. Luckily, while the rules vary in phase number, in all cases the demarcation lines between schemes have a consistent logic such that phases for one RoT are merely combinations of phases of other schemes.

The approximate distribution of the eleven RoT are displayed in Figure 2.3. For ease of discussion, they have been grouped by scheme (number of phases) to reflect their commonalities. Two of the RoT discussed—Heijstek and Papatheocharous—split up the first SRDR phase, Requirements Analysis, and allocate some effort to a "Planning" phase. However, the work in this phase so closely matched the SRDR's Requirements Analysis that we combined it all into the one phase. Thus, this "planning" phase is not depicted in the chart.

The top group of the chart displays the rules with either five or six phases. Yang, Papatheocharous, Heijstek, and Borysowich have similar values for the first four phases. Heijstek and Borysowich maintain six phases, and are thus the closest match to the six-phase format of the SRDR database. Yang and Papatheocharous do not allocate effort to a specific Phase 6, instead combining Phase 5 and 6 (shown as a blend of red and orange).

The middle group consists of three RoT by Ambler, Zelkowitz, and Sommerville. Each uses a four-phase scheme and keeps phases 1 and 2 separate. In this scheme, the SRDR phases 4, 5, and 6 are combined (hereafter phase 4, and shown by the autumnal brown–a blend of yellow, orange and red). Though grouped in this manner, Ambler appears a strong outlier due to the rule's large allocation to phase 3 (Coding) and small allocation to the combined phase 4. Comparatively, Zelkowitz and Sommerville are very similar, with Sommerville shifting effort from phases 1 and 4 to phase 2. Moving further down, Brooks stands abandoned in his own scheme. While he does

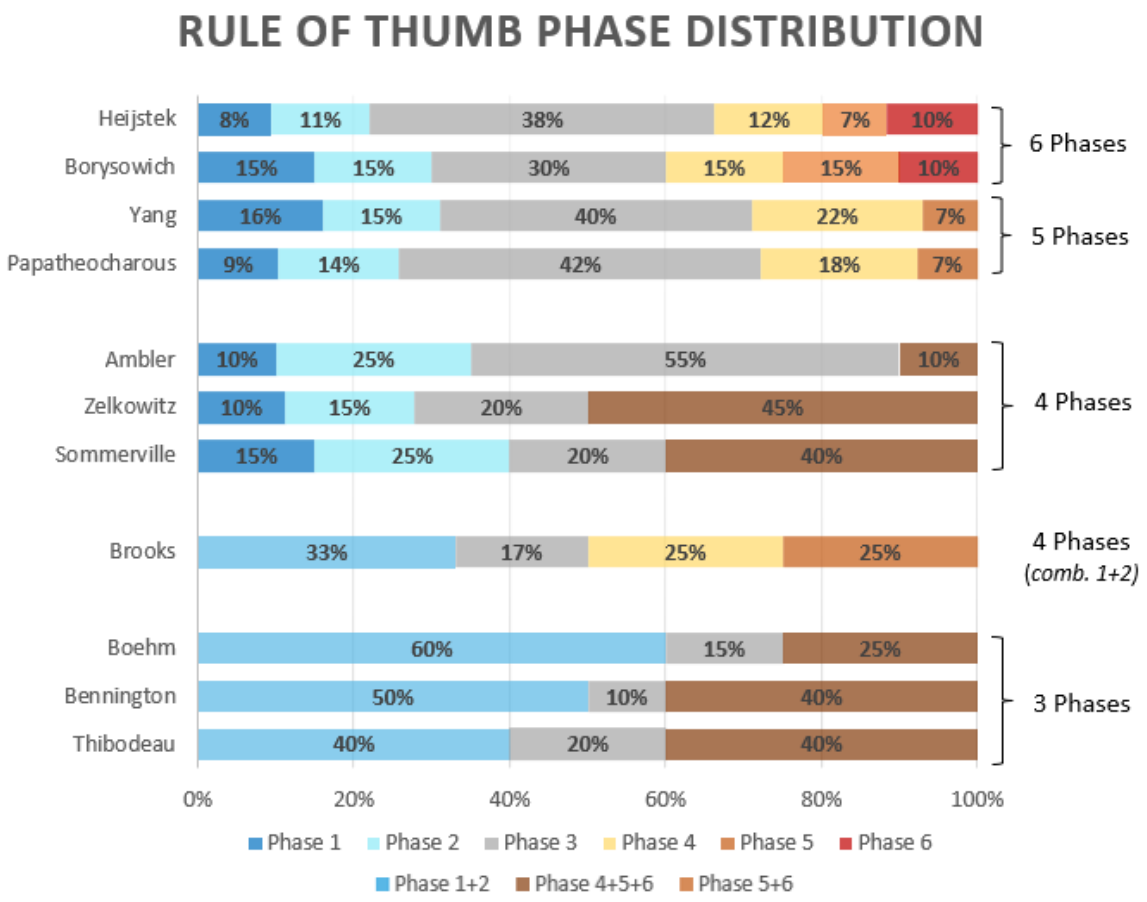


Figure 2.3: Grouped Rule of Thumb Distributions

have four phases, he merges phase 1+2 (shown in the blended color of baby blue) and keeps two phases separate after the Coding effort. This makes the rule dissimilar to the prior three rules, but also cannot fit into the last scheme for rules with three phases.

The final scheme shows three rules with very simple phasing by Benington, Boehm, and Thibodeau. Like Brooks before, they combine the first two phases and therefore place a large emphasis on the requirements and design phase before starting Coding. At 60% and 50%, Boehm and Benington allot a greater percentage to phases 1+2 than any other RoT. They also place a lower emphasis on coding (gray, phase 3) compared to other RoT, which average between 20-40%. They pose as stark contrast

to the schemes in the first cohort.

Ambiguity arises when comparing *Testing* effort among researchers since the SRDR describes its Phase 3 as "Coding & Testing". Where is testing being done? What kind of testing is performed in Phase 3 vs. the other two test phases? Based on the descriptions of phases in the SRDR guide and the description of phases for other researchers, the SRDR phase 3 is most closely aligned with pure coding/construction (Lanham et al., 2018). Formal testing, quality analysis, and integration testing with other systems is done in Phases 4, 5, and 6. Though small discrepancies may arise, this paper considers the SRDR Phase 3 to consist only of coding/construction, and assumes formal testing only occurs in Phases 4, 5, & 6.

Researchers define the initial project effort before the coding stage very differently. Though the SRDR scheme defines two phases (Requirements Analysis and Architecture & Design), the literature varies. For instance, the Heijstek RoT splits the work prior to coding among three phases: Planning, Requirements Elaboration, and Analysis & Design (Heijstek and Chaudron, 2008). Sommerville defines two phases, Specification and Design (Sommerville, 2016). In total, four RoT combine all initial work into one phase, five RoT use two initial phases, and two RoT use three initial phases.

It is clear from a survey of these phases that the literature is quite disparate. There are a plethora of guidelines on how much effort is appropriate for each phase, though the guidelines have been constructed from varying datasets, application types, software sizes, and decades of development and expert opinion. The differing definitions of phases included in the development process muddles the analysis of each study. There has been some empirical work to show mean phase distributions, but no comprehensive phase or effort analysis on the SRDR dataset has been performed (the 2011 Tan study should be dismissed). Whether the RoTs are merely context de-

pendent or errant conceptually, one should not have faith that they are generalizable or that they could be prescriptive to a specific context such as the DoD.

2.4 Phase Relationship Analysis

Determining which RoT best matches available data serves to provide a heuristic to future program managers. Once early researchers focused on phases, they began to consider how the phases are interrelated, and how they might impact program success, costs, and overall schedule. Though partially discussed above through Norden, Putnam, and Brook's research, the following researchers sought to dive deeper than a surface-level understanding of phase relationships and schedule.

In early phase research, Thibodeau & Dodson attempted to reflect interrelationships between phases (1980). They postulated that there was a measurable trade-off between Analysis/Design and Coding/Testing along isoquant lines, similar to a Cobb-Douglas function. In such a model, if design effort increased (the x-axis), then coding effort decreased (the y-axis). Unfortunately, their dataset was too small to produce significant results.

The concept was later tested by Yiftachel et al., who studied the choices present in deciding when to move between phases (Yiftachel, 2011). The researchers tasked undergrad students with building a program and reviewed their effort and schedule across all phases. They found that isoquants for development exist, where individuals could arrive at the same end result through different ratios of design and coding effort. Students transitioned between Design and Coding when the marginal benefit to move on outweighed the marginal benefit to fully flesh out the current phase. When challenges occurred in the new phase (e.g. coding), the challenges of persevering in the face of missing data would decrease the marginal benefit and it would then make sense to revisit old phases (e.g. Requirements Analysis) to build the foundation and

architecture before continuing coding.

There are several implications of the research by Thibodeau & Dodson and Yiftachel et. al. First, the rigidity of the Traditional process is not necessary and can be detrimental to overall schedule, and can move teams into periods of sub-optimal productivity. Second, more effort in initial phases does not necessarily equate to greater efficiency later on, as that effort may be a result of the sub-optimal state of productivity, and not concerted and meaningful up-front investment in design.

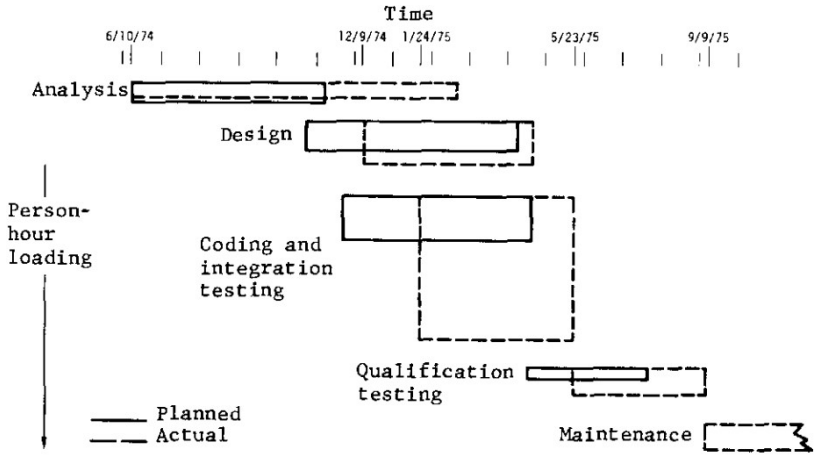


Figure 2.4: Example Planned vs Actual Schedule (Thibodeau, 1980)

Thibodeau & Dodson (2008) also explored the effects that earlier phases can have on later ones. They considered examples where there were unexpected delays in the completion of earlier phases. If the start of later phases were not adjusted to accommodate schedule extension in an earlier phase, then the total effort required would increase *in addition to* total schedule. This relationship is described in Figure 2.4. The solid lines reflect the initial phase plan in the common form of a Gantt chart. The dashed lines reflect the end result. The Analysis phase was extended, but the schedule and effort for the following phases were not adjusted accordingly. The decision to hold the phases to their start time put stress on the team. The result was schedule growth in subsequent phases (measured horizontally), but also, importantly,

a disproportionately large increase in effort (measured vertically). A shortcoming of fluid phases as identified above is that programmers may haphazardly design or define aspects that should have been taken care of in previous phases, instead of marking the issue and allowing the program manger to revisit the early stages (Thibodeau and Dodson, 1980; Yiftachel, 2011). There are several implications. First, completing phases may be important after all. Second, as predicted by Brooks, total effort may grow disproportionately when problems emerge in the initial design phase, and when schedule is compressed or constrained.

Boehm and Basili also honed in on the implications of problems in the earlier phases. They found that the *Pareto Rule* applied strikingly well to software development (Boehm, 1987; Boehm and Basili, 2001). That is, 80 percent of the problems in a project stem from 20 percent of the code. Moreover, they found the majority of the defects to be specifically due to poor requirements definitions or design work. This finding was later affirmed by Jones (cited in Everett article), who found that 85 percent of development defects were introduced in the design phase (Everett and McLeod Jr., 2007). Separately, Taipale et. al. interviewed 40 industry experts to understand what factors practitioners perceived as contributing to schedule overruns (Taipale, 2006). Consensus pointed towards communication and interactions during the design phase as the primary factors in schedule slips.

The interrelationships between phases are complex, but they point toward early phases are significant toward overall program success. Thibodeau and Dodson found that a delay in an early phase will increase the effort and time to complete later phases. Boehm and Basili identified which phase was most likely to have a delay, the Design phase. Multiple researchers argue that improving the work done in the early phase will improve results overall, both in terms of schedule and cost.

2.5 Conclusion

Rules of Thumb have flourished in the literature. However, validation and comparative assessment have been sparse. The DoD has done almost nothing to consider their relative merit for its specific context. As for the meaning or importance of any given phase for performance, it has received some attention in the commercial literature, but none within the DoD, which has placed its efforts on directly measuring cost without considering schedule as a factor. What the research in the commercial sector suggests is that schedule is a factor. In particular, errors and delays in early phases affect the completion of later phases. moreover, there are disproportionate effects; early issues cannot be efficiently solved by simply adding manpower and keeping schedules constant. From this assessment of the literature, a research agenda in the context of DoD data is clear. First, to guide the practitioner towards a modern, DoD-applicable rule of thumb. Second, to improve our understanding of the relationships between phases and schedule.

III. Methodology

3.1 Overview

This section reviews the methods used to analyze the SRDR data. There are multiple methods used, unique to each research question. To assess the merits of the rules of thumb, we applied three different tests for the sake of robustness. With descriptive statistics, we observed general phasing trends across multiple categories of projects. We then used t-tests to determine the effects of early effort allocation on schedule growth. Before we can begin analysis, we need to understand the dataset and how to cull to the usable data entries for the purposes of our research.

3.2 Data

The research data comes from the Software Resources Data Report (SRDR) compilation, a repository of software contracting reports for the DoD dating back to 2001. Each report, or row in the database, is an individual Computer Software Configuration Item (CSCI). Each CSCI is a base component of a larger system, performing a specific task or set of tasks (AFCAA; NCCA, 2008). For example, a radar software contract may include 6 CSCIs, each of which is needed to make the radar system operate as a whole. In this thesis, we will use the terms "report" and "project" interchangeably to describe CSCI rows in the database. The SRDR database catalogues 150+ metrics across 5074 DoD software reports from approximately two thousand system-level projects (a.o. August 2021). Most of the projects are from the three main branches of Army, Navy, and Air Force, and represent multiple military application types. Contractors are required to submit initial, interim, and final reports on their software progress to the SRDR database, leading to multiple rows for each unique CSCI.

The total data set included 5074 reports. Only a portion of these are usable for the purposes of this research. We are primarily interested in patterns that occur in the Air Force, Army and Navy. Therefore we removed 103 reports pertaining to the Missile Defense Agency, 176 for the US Marine Corps, and 17 for the DoD. Brief analysis showed that these other services had very different phasing distributions, and exploring the reasoning for those patterns is outside the scope of our effort. Next, we removed reports that were missing a quality tag indicator which is called the "Validation and Verification (V&V)" tag. This tag is applied by the DoD data repository center, where the data is validated and verified as satisfactory for analysis if it contains most of the entry fields and is generally usable for future estimates and cost models (Lanham et al., 2018). It should be noted that while there are guidelines on how/when to apply the quality tags, there is some room for human judgement. Many projects do not meet the criteria for a "Good" V&V tag, leaving only 1456 projects for credible analysis. Due to the nature of our research, we are only interested in initial and final estimates; therefore, we further reduced this dataset to include only those entries designated a unique initial or final report. We were not interested in interim actions or partially complete projects. This criteria restricted the number of projects to 1128. Full breakdown of the exclusion process for the first research questions is in Table 3.1. These 1128 reports will allow us to study the effort allocation for all initial and all final reports and assess which RoT fit the resulting distributions.

Table 3.1: Steps of Exclusion for First Tests

Exclusion Criteria	**# Removed**	**# Remaining**
Initial Data Set	0	5074
Data not belonging to Air Force, Army, or Navy	301	4773
Data not matching "good" V&V quality tag	3317	1456
Data not considered an Initial or Final report	328	**1128**

To determine schedule growth and phase estimation accuracy, we must be able to compare initial estimates to final results for the same project. An additional set of exclusion criteria were applied to only consider projects that have a corresponding data pair of initial *and* final. This requirement culled the tally of usable report pairs to 377. Since we focused on schedule at this part of the analysis, we also eliminated all projects tagged as an "Impossible Schedule." These tags are applied by the DoD data repository center when the start and end dates were improperly entered, resulting in a zero value or negative schedule for completion. Finally, we eliminated 13 reports that were missing phase values for both Phase 1 and Phase 2, since the literature suggested that these phases were important for understanding growth and changes overall. As long as a report included at least one or the other, we included it. Table 3.2 depicts the removed and remaining report values for each exclusionary step.

Table 3.2: Steps of Exclusion for Second Tests

Exclusion Criteria	**# Removed**	**# Remaining**
Starting point from First Test exclusions		1128
Data without an associated pair	751	**377**
Pairs without an "Impossible Schedule" tag	56	**321**
Missing effort for both Phase 1 and Phase 2	13	**308**

3.3 Overall Distribution Patterns

To start, we looked at the general distribution of effort in each phase, as well as the change from initial to final. Each phase contains a measure of the hours required to complete it. The Hours variable serves as a measure of Effort, the traditional approach to studying the effects of software. The hours for each phase were converted into a percentage of the *total* programming hours for the project. The phasing percentages were calculated as the mean effort percentages for each phase, looking only at final projects. Initial projects were ignored for this view as we wanted to see what the

average development process produced. The distribution was evaluated for multiple categories, such as Service and Operating Environment.

Using the smaller paired dataset, we found the change in percent effort from initial to final for each project and averaged the results by phase. This gave us an idea of how much our phasing distribution changed between our initial estimate and our final result. Because the size of a project is a common cost relationship (larger software programs take longer to develop), we also measured the change in average software size by phase. Together, these two calculations indicated how well we predict our phasing and how much effort changes impact software size.

3.4 Rules of Thumb

After evaluating the overall distribution patterns, we compared them to the rules of thumb. The mean percent effort for each phase in the distribution was compared to the respective percentages asserted by each rule of thumb. This analysis did not require data pairs, so the larger dataset of 1128 was afforded to get a better sense of how projects compared to the rules of thumb.

Although the SRDR is designed for six phases and receives some data in that scheme, most of the projects are submitted in different phase schemes. Contractors

Inclusion Criteria	**# Remaining**
Data tag of "Good", and an Initial or Final report (see Table 3.2 for breakout)	1128
i) Data with values for all six phases	210, 19%
ii) Data with *at least* values for Phases 1, 2, and 3	775
iii) Data with *at least* values for Phases 1, 2, 3, and 4-6 combined	723
iv) Data with *at least* values for Phases 1-2 combined, 3, and 4-6 combined	878

Table 3.3: Aspects of Inclusion for the Rule of Thumb Analyses

regularly employ different phase models internally and thus, we see several alternative patterns in the data. Table 3.3 shows the frequency of each pattern of submission. Only 20 percent of the submitted reports use all six phases. Some are partial. Many more follow recognizable patterns mirroring the other schemes seen in the literature.

Given the multitudes of phase schemes both in the above RoTs and in the SRDR submissions, only small cohorts would match any given RoT. However, despite there being a different number of phases among the rules, the demarcation lines of phases aligned conceptually. By way of analogy, there is a North Carolina and South Carolina, but some people speak of "The Carolinas." Nobody disputes what the geographical outline of that conceptual entity is. In a similar fashion, "Requirements Elaboration" and "Design" are phases that some people treat as a singular phase called "Analysis." In order to test the RoTs against more of the DoD programs, and thus arrive at more generalizable finding, we could take our often more precise phases in the SRDR and combine them as necessary to match to simpler RoTs. A scrutiny of the descriptions of these phases gave us faith of the appropriateness of these simplifying transformations. That being said, there is room for error. The delineation of each rule based on our SRDR definitions is a human process, and thus a candidate for sub-par analysis. We believe our assumptions and categorizations are reasonable and appropriate, but we are not foolhardy enough to consider them statistical.

The methodology we applied, then, is one where we take the same large dataset of 1128 projects and combine their phases in various ways to test the whole set against each RoT. We were able to test all 11 RoTs by 3 simple combinational forms, which we call test sequence A, B, and C. Figure 3.1 shows those sequences.

Four rules use five or six phases. Heijstek and Borysowich used a six-phase scheme, while Yang and Papatheocharous treated the last two SRDR phases (5 and 6) as a single phase. While reviewing the data, we found inconsistencies in reporting for the

A	1	2	3	4	5 + 6
B	1	2	3	4 + 5 + 6	
C	1 + 2		3	4 + 5 + 6	

Figure 3.1: Phase Combinations

last SRDR phase. To limit testing issues, we transformed our 6 phase SRDR data into the 5-phase scheme. This is shown as **"A"** in Figure 3.1. The RoT by Heijstek and Borysowich were summarily transformed.

Four rules of thumb used 4 phases. Ambler, Sommerville, and Zelkowitz matched the first three SRDR phases, then combined 4 thru 6 into a single phase encompassing all Testing, Integration, and Transition activities. Accordingly, we transformed our SRDR into the same 4 phase format, test sequence **"B"**. Brooks, on the other hand, kept his last two phases separate and instead combined the initial effort into a single Analysis and Design phase. We considered making this a separate test sequence where phases 1+2 were combined, 3 and 4 were separate, and phases 5+6¬ were combined. To simplify SRDR transformations and comparisons between phases, we instead combined Brook's last phases to align with test sequence **"C"**, elaborated below. Finally, three RoT use 3 phases: Boehm, Benington, and Thibodeau. The phase used are effectively Analysis & Design, Coding, and Test & Integration. Analysis & Design mirrors our phases of Requirements Analysis and Architecture & Design. Test & Integration mirrors our phases Integration, Qualification Test, and Development Test & Evaluation. Thus, we transformed the SRDR data into the same format by merging Phase 1 + 2 and Phases 4 thru 6. We call this test sequence **"C"**.

3.5 Normality

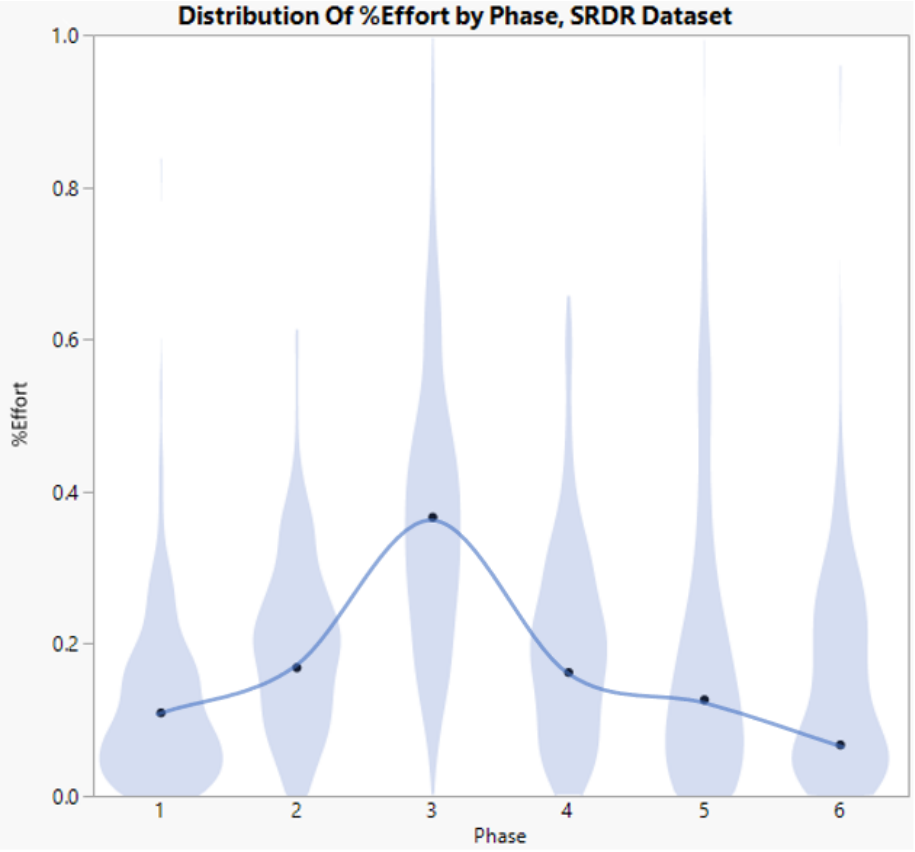


Figure 3.2: Phase Effort Distribution, Overall Dataset Final Values

Figure 3.2 was produced as a cursory look at the normality in the phase distributions. We then used the Shapiro-Wilks normality test to evaluate if the effort distributions of each phase matched a normal profile. The null hypothesis, H¬0, assumes normality. A small p-value less than 0.1 rejects the null value and indicates likely non-normality. The Shaprio-Wilks test results are displayed in Table 3.4. The data failed normality for each phase, rejecting the null hypothesis. However, the central limit theorem can be invoked given the large N-values for nearly all tests, even though some distributions appear non-normal (e.g. log-normal) in the case of the Phase 1 and 5 distributions.

Table 3.4: Normality Tests for Dataset

	Phase	Shapiro-Wilk Value (W)	p-value
Initial	1	0.687	<0.0001*
	2	0.941	<0.0001*
	3	0.938	<0.0001*
	4	0.918	<0.0001*
	5	0.700	<0.0001*
	6	0.606	<0.0001*
Final	1	0.702	<0.0001*
	2	0.818	<0.0001*
	3	0.958	<0.0001*
	4	0.900	<0.0001*
	5	0.722	<0.0001*
	6	0.606	<0.0001*

Figure 3.2 shows some of the variance in each phase, but we utilized a box-and-whisker plot to get a better sense of the variance. Figure 3.3 shows the percent effort in each phase, with each whisker ending at the 75th percentile. The figure demonstrates the large variance in certain phases. The largest inter-quartile range is for phases 3 and 4, with the smallest range for phase 6.

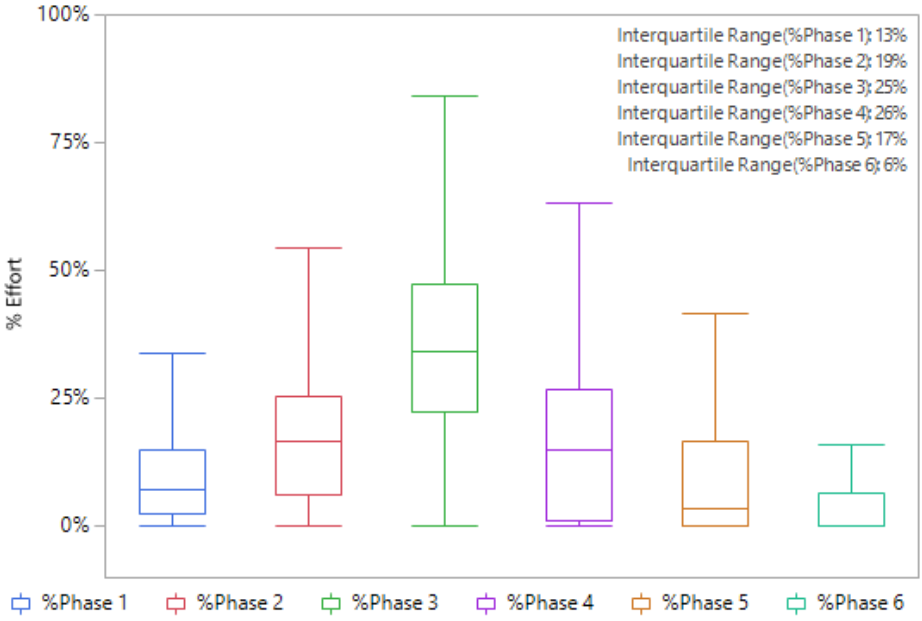


Figure 3.3: Variability by Phase

3.6 Overall Distributions

An ANOVA can test individual phase means against the expected RoT value for each phase, but cannot test how well the full RoT matched the full phase data. In Figure 3.4, the closer the rule is to the data at each phase, the smaller the absolute deviation will be (dashed orange and blue lines). Four of 5 phases might be very close, but the fifth phase could be drastically different and make the rule as a whole a poor fit. The two rules displayed are both close to the data and far from the data, but at different phases. Which one is closer, on average? Which rule is a better fit? To compare the phase *distribution* as a whole to each RoT, as in the figure, other tools are available.

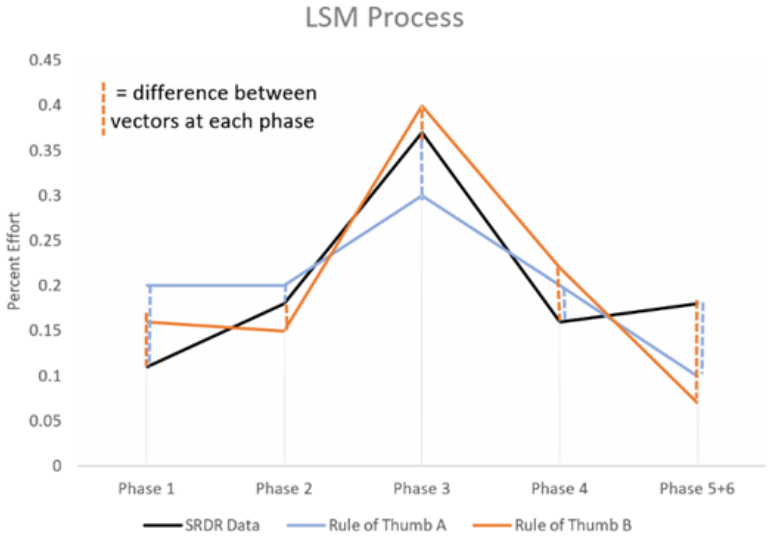


Figure 3.4: Example of Deviation for Each Phase

The challenge of testing the goodness of fit of something that has multiple phases is that the test is a multivariate question, not a univariate one. Each phase is its own variable. We cannot simply test phase 1 of the data against that predicted by the RoT for that phase, and so on. Luckily there are tools for this. Generalized versions of the standard t-test (ANOVA) can be used for this multivariate context. We employ two: Least Squared Means (LSM) and Hotelling's T^2 test. These tools consider each

distribution of effort as a vector, then calculate the absolute deviation of the RoT vector from the data vector.

The first method, the LSM process, uses a similar technique to ordinary least squares regressions. It uses squared differences to measure how far a rule is from the data at each point. First, we squared the difference between the SRDR data phase 1 mean and the RoT phase 1 mean, then repeated for each phase as appropriate for the scheme. (See equation 3.1). These squared differences were then summed, and the square root was taken of the final sum. This value essentially served as a magnitude of divergence; the greater the magnitude, the greater the mean phase values differed from the RoT phase estimates. This is visually represented in Figure 3.4, where the dashed vertical lines represent the difference between phase means. For our results, a smaller LSM magnitude indicates a better fit for the RoT data.

$$\textit{LSM Magnitude of Difference} = \sqrt{\sum_{i=1}^{i}(\bar{x}_{data} - \bar{x}_{RoT})} \textit{ for } i \textit{ phases} \tag{3.1}$$

The second comparison method used Hotelling's T^2 test, a multivariate form of the standard t-test (Schumacker, 2016). In technical terms, Hotelling's test compares the pairwise means of multiple points within each vector. For the SRDR data, we can consider the phasing distribution to be a vector, with the percentage of effort in each phase a single point on the vector. Similar to the t-test, the resulting statistic is converted into an F-test value and evaluated against the F-table for significance. The Hotelling's analysis was performed in R.

The necessity of this test can be explained through the following analogy. Consider the test of whether a piece of currency is genuine or counterfeit. There might be four characteristics to test for: color of the ink, shimmer of a metallic strip, etc. Some deviation in any of these is acceptable since printing devices have some variance/-tolerance. However, deviation too far for a single characteristic would, and should,

trigger a counterfeit alarm, even if the other three are spot-on. A rather good counterfeit would get rejected for a singular slip up. So it is with the Hotelling's test, and it represents the most robust of the tests we can apply here.

Hotelling's test statistic, T^2, is derived from a general t-distribution:

$$T^2 = n(\bar{x} - \mu_0)\mathbf{S^{-1}}(\bar{x} - \mu_0) \tag{3.2}$$

where

- n is the sample size
- $\bar{x}$ is a $1 \times p$ matrix of variable values for p phases (the effort percentages in each phase)
- μ_0 is a $1 \times p$ matrix of the parameter values to test the means against (the Rule of Thumb values)
- S^{-1} is the inverse of the sample's covariance matrix

This statistic is then converted via equation 3.3 into a F-statistic and evaluated within the F distribution.

$$F_{p,n-p} = T^2 \frac{n-p}{p(n-1)} \tag{3.3}$$

We applied these techniques to test the whole data set. We followed that test with a series of tests of various categories of the data, to determine if the RoT were a better fit for given subsets of the data. We looked at Service, Application Domain, Super Domain, Operating Environment, Development Style (Traditional vs. Agile), and by the size of the code. Within software literature, each of these categories have been shown, in varying degree, to have distinct qualities, and are thus worthy of separate analysis.

The software code size is usually measured in ESLOC, or Equivalent Source Lines of Code. ESLOC normalizes the types of code used in a project, since one coding language can require 3 lines of code to perform a function while another language may only take one line for the same function. The lines of code are also normalized to account for non-functioning lines, such as descriptive comments, as well as how much of the code is made up of new, modified, or reused code. To evaluate the size, we broke out the ESLOC values into six separate bins. The bin sizes used in our research are loosely based on other research and the COCOMO II analysis platform, and separate effectively separate analysis between small, medium, and large projects (Papatheocharous et al., 2017; Yang et al., 2008).

Regarding application domain type, previous studies claim that application type has a limited effect on development (e.g. radar software vs. enterprise systems) (Amato, 2021). The SRDR categorizes each project into one of 14 "application domains," then rolls each domain into one of four "super domains." Both application domain and super domain will be analyzed within this research. The applications and domains evaluated are described in Figure 3.5.

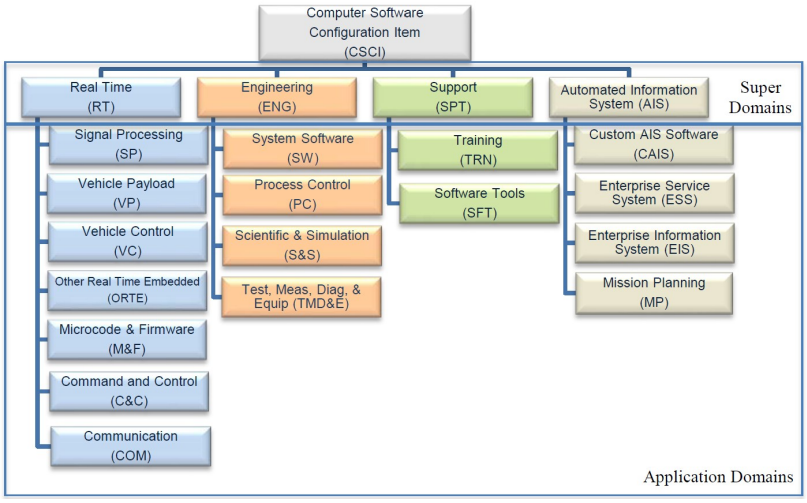


Figure 3.5: Application Domains and Super Domains within SRDR (Lanham et al., 2018)

3.7 Phase Relationships & Schedule

The second area of analysis sought to elucidate the nature of the relationships between phases and schedule/cost outcomes. This section evaluated research questions 3, 4, and 5, looking to answer inquiries like "Does a higher-than average effort in Phase 2 reduce the overall schedule growth?" Because this analysis needed to assess changes from the initial estimate to the final result, the smaller dataset incorporating paired projects was used.

Using a post-hoc approach, we identified high and low schedule growth projects, then used t-tests to see how the growth was affected by the effort distribution. The dependent variable for the schedule tests is the percent schedule growth from initial to final. The change in schedule is measured simply by the difference between the estimated months and the actual months, normalized by dividing this difference by the total months of the initial estimate. This measure produces a percentage increase/decrease from the initial estimate. The mean percent increase was found, then used as a dividing line to split the data into two cohorts for projects above and below the mean schedule growth. The same process was repeated with median values. The independent variables are the percentage of effort spent in each of phases 1 and 2, and 3. Differences in the last three phases were initially ignored since we are primarily interested in early phase effects in accordance with the literature. We will consider them in our robustness checks.

We used the t-test to evaluate the effort allocation of projects with a greater than average schedule growth against those with a less than average schedule growth (or possibly a schedule decrease). Given the irregularity of the data, we applied a t-test with $\alpha = 0.10$ as a guide for significance. The null assumes that the effort allocations between the two cohorts are the same. We conduct this test for the overall dataset, and then for the same subcategories identified above: Service, Super Domain, Appli-

cation Domain, Operating Environment, Process, and Size. Statistically significant results for each phase would indicate that the effort allocation in Phase X was *not* the same between the two cohorts of schedule growth, our DV. For example, a statistically significant Phase 1 test result for the Army would state that Army software projects with a lot of schedule growth allocated their Phase 1 effort differently than those Army projects with less than average schedule growth. As with the RoT tests, multiple categorical comparisons were considered, such as Service and software size.

IV. Results

4.1 Overview

This chapter presents the results from multiple tests on the data. The first section uses three separate approaches to test which Rule of Thumb best fits the data, using the final phase values. The second section compares the initial estimates to final outcomes for the projects and attempts to answer how well we get our initial estimates right. Finally, we grouped the data into cohorts with above or below average schedule growth to identify how the distribution of early effort relates to schedule.

4.2 Rules of Thumb

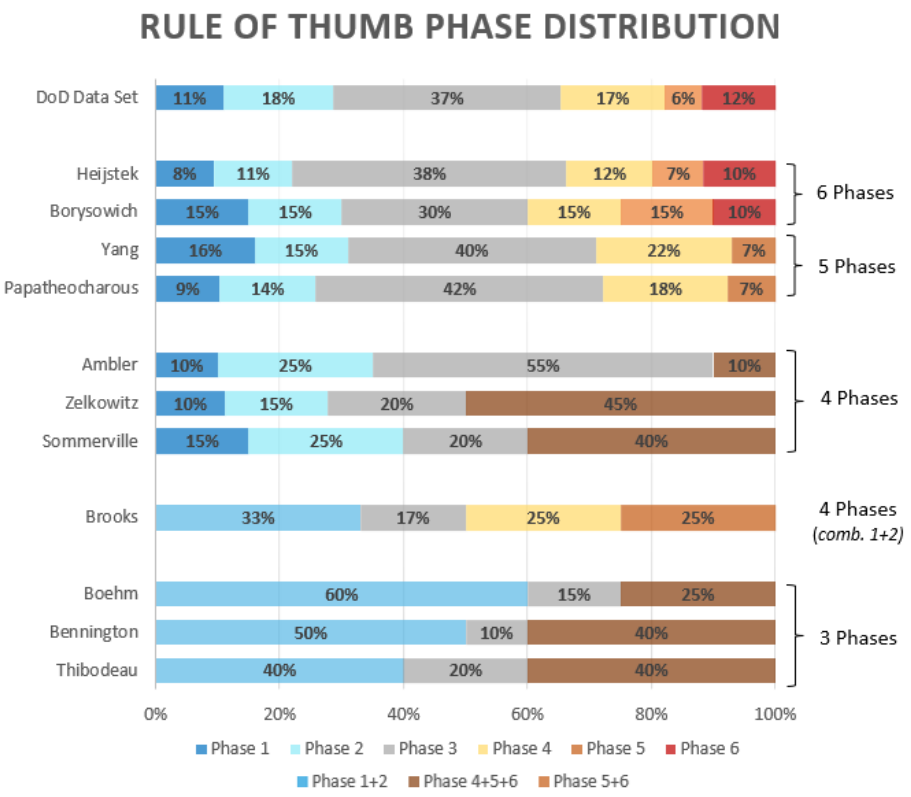


Figure 4.1: Final Phase Distributions (Actuals) Compared to Rules of Thumb

In Figure 4.1 we show again the RoT grouping diagram from Chap 2. At the top we have added the mean phase efforts seen in the SRDR database. It provides an initial overview of the similarities of RoTs to the SRDR data. A few observations are readily apparent. Yang, Papatheocharous, Heijstek, and Borysowich roughly visually resemble the SRDR effort. Heijstek nearly matches the phase 3 percentage, despite that his phase 1+2 effort are larger than that of the dataset. Though the Yang RoT see divergence from the dataset in the latter phases, it is remarkably similar to the data for phases 1 through 3.

The second group breaks further from the dataset values, with none of the rules appearing a good match to the data. Ambler's allocation of the largest proportion of effort to coding, 55 percent, is not reflected well. Sommerville and Zelkowitz allocate just 20 percent of effort to the Coding phase, an allocation which is not reflected in the data either. This may partially be due to their studied datasets, which were focused in the industrial sector and are not a 1:1 match to the SRDR data. On the good side, Ambler and Zelkowitz' 10 percent effort in phase 1 is just 1 percent off the actuals value. The last grouping shows schemes that appear radically different from the actual data. Even though the simpler 3-phase scheme allows for more leeway in the comparison to the dataset, the values are wildly disparate from the dataset. Boehm, Benington, and Thibodeau focus about half their effort on the first two design phases, but bears little resemblance to the scope of actuals at 29 percent. Each rule allocates between 10-20 percent to Coding, but the dataset mean is 2-4 times larger.

The next table, 4.2, shows the mean final effort distribution for all categorical filters. The color scale visually describes where the distribution of effort is focused for each category, with darker colors highlighting more effort and lighter colors reflecting a lower effort phase. The darker colors overwhelmingly spotlight phase 3, where the bulk of coding is and the highest average effort. As expected, the distribution tails

off in the last three categories—one can imagine a traditional Rayleigh curve overlaid on each row of the table.

We left this table as a descriptive guide only, due to the sheer number of possible ANOVA comparisons to test significance of differences between phases and categories. We feel confident in making two observations. One, the relative distribution between phases does not change considerably between categories. Phase 1 effort remains close to the dataset average of 11%, Phase 3 effort tends towards the average of 37%, and so on. There are a few drastic exceptions, like the domain TRN, but they are not the norm. Two, because the distribution remains relatively constant regardless of category, we are encouraged that subsequent RoT and schedule results will be generally applicable to all categorical filters.

		Final Percent Effort, Mean Values					
	N	Phase 1	Phase 2	Phase 3	Phase 4	Phase 5	Phase 6
Full Dataset	722	11%	18%	37%	17%	12%	6%
Service							
Air Force	230	8%	19%	40%	19%	8%	6%
Army	216	16%	18%	33%	14%	17%	3%
Navy	276	11%	18%	36%	17%	11%	7%
Super Domain							
AIS	48	9%	21%	44%	16%	6%	4%
ENG	114	9%	21%	35%	18%	11%	6%
MS	26	18%	22%	30%	15%	6%	9%
RT	534	12%	17%	37%	17%	12%	6%
Application Domain							
C&C	164	13%	16%	42%	12%	11%	6%
CAS	15	11%	23%	41%	16%	6%	3%
COM	68	7%	24%	32%	21%	11%	4%
M&F	1	7%	26%	33%	30%	0%	4%
MP	33	8%	20%	46%	15%	7%	4%
PC	1	7%	18%	60%	5%	4%	6%
RTE	119	13%	18%	35%	18%	11%	5%
S&S	33	10%	18%	32%	17%	15%	8%
SP	77	12%	18%	33%	22%	10%	5%
SS	60	9%	23%	36%	21%	5%	6%
TMDE	20	5%	18%	37%	14%	22%	4%
TOOL	16	12%	27%	27%	23%	5%	6%
TRN	10	26%	16%	34%	4%	7%	13%
VC	62	13%	14%	35%	14%	17%	7%
VP	43	10%	12%	35%	19%	20%	4%
OE - Summary							
Air Vehicle	268	9%	16%	39%	17%	13%	7%
Sea System	78	9%	18%	37%	17%	18%	1%
Space System	48	10%	14%	39%	14%	8%	14%
Surface	339	15%	21%	34%	17%	10%	5%
Process							
Agile	123	22%	13%	41%	10%	9%	5%
Traditional	595	9%	19%	36%	18%	12%	6%
Initial Size							
<5K	104	13%	14%	34%	20%	15%	4%
5-20K	188	11%	19%	37%	17%	12%	4%
20-50K	174	9%	20%	36%	18%	11%	5%
50-100K	113	10%	17%	39%	15%	10%	9%
100-250K	105	14%	19%	35%	14%	11%	7%
>250K	38	15%	20%	36%	15%	9%	6%

Figure 4.2: Final Percent Effort Allocation for Each Phase

4.2.1 Least Squared Means

We used two methods to go beyond this eyeball method and to test the actual fit of the RoT. We apply these tests to the overall dataset as well as to the previously identified subcategories of interest.

The first method to evaluate RoT fit is the Least Squared Means method. This is a multivariate method that measures the average difference between each phase mean and the RoT values. The closer the LSM value gets to zero, the better fit that rule is to the data. The LSM method is only an approximation for matching rules of thumb since it ignores the variance in each phase and only compares the mean value. We combined phases 4+5+6 to enable comparison across the greatest number of rules of thumb while still maintaining the granularity of keeping phases 1, 2, and 3 separate.

Table 4.1 shows the results of the LSM method. A smaller magnitude indicates less divergence from the data and thus a better fit. The rows are sorted in order from least to greatest average magnitude according to the Full Dataset column; this sorting is consistent for each successive table. Yang had the closest overall magnitude for the full dataset, the Air Force and the Navy. Though not the best fit overall, Army was better described by Borysowich than Yang. The runner up was Papatheocharous. The worst matches to the data were by Ambler and Benington.

In Table 4.2, the same model is run, but evaluated according to Operating Environments. These partially track with Service (e.g. all Space is under Air Force, and all Sea is under Navy). The rows are sorted in the same order as the first table, from smallest average magnitude to largest for the overall dataset. The results match the previous findings, with Yang, Borysowich and Papatheocharous the closest match to the data. Looking horizontally, it would appear that all of the RoT do worst at predicting Space.

Table 4.1: LSM Magnitude for Full Dataset and by Service

Rule of Thumb	**Full Dataset**	**AF**	**Navy**	**Army**
N	*1124*	*346*	*339*	*439*
Yang	**0.0825**	**0.0980**	**0.0970**	0.0860
Borysowich	0.1225	0.1594	0.1273	**0.0616**
Papatheocharous	0.1364	0.1463	0.1517	0.1304
Sommerville	0.1975	0.2311	0.1876	0.1637
Heijstek	0.2025	0.2140	0.2163	0.1855
Thibodeau	0.2112	0.2486	0.2040	0.1594
Zelkowitz	0.2145	0.2417	0.2005	0.1995
Brooks	0.2592	0.2922	0.2412	0.2337
Ambler	0.3082	0.2818	0.3320	0.3314
Benington	0.3850	0.4217	0.3826	0.3240
Boehm	0.3906	0.4217	0.3982	0.3262

Table 4.2: LSM Magnitude by Operating Environment

Rule of Thumb	**Surface**	**Space**	**Sea**	**Air**
N	523	61	114	437
Yang	**0.0860**	**0.1342**	**0.0894**	**0.0927**
Borysowich	0.0872	0.1913	0.1407	0.1556
Papatheocharous	0.1304	0.1838	0.1435	0.1456
Sommerville	0.1806	0.2379	0.2093	0.2241
Heijstek	0.1913	0.2454	0.2102	0.2112
Thibodeau	0.1849	0.2687	0.2276	0.2454
Zelkowitz	0.2112	0.2358	0.2209	0.2328
Brooks	0.2619	0.2698	0.2650	0.2728
Ambler	0.2943	0.3641	0.3127	0.3212
Benington	0.3432	0.4532	0.4047	0.4250
Boehm	0.3335	0.4819	0.4145	0.4384

Table 4.3 provides another filter on the data using Super Domains. This view allows us to see if the rules are sensitive to common technical ends to which the software is employed. RT means Real-Time, the most constrained and difficult software to develop. It is also the most common of programs studied in this report, with N=830. ENG is Engineering, which may take RT output and further process for human interaction. AIS is Automated Information System, which provides information processing services. MS is Mission Support, the least constrained, and has wide flexibility in output and testing. Of existing RoT, Yang again had the closest magnitude to the Services, followed by Borysowich and Papatheocharous. This third view shows that the same rules are following closely to the data; if this holds in other categorical filters, then the Yang rule may be consistently the best fit overall. One qualification to this repeat finding is that the MS Domain was best described by Borysowich.

Table 4.3: LSM Magnitude by Super Domain

Rule of Thumb	**RT**	**MS**	**ENG**	**AIS**
N	*830*	*36*	*179*	*79*
Yang	**0.0825**	0.1020	**0.1241**	**0.0970**
Borysowich	0.1225	**0.0707**	0.1400	0.1594
Papatheocharous	0.1364	0.1257	0.1783	0.1131
Sommerville	0.1975	0.1924	0.1749	0.2709
Heijstek	0.2025	0.1738	0.2437	0.1691
Thibodeau	0.2112	0.1913	0.1944	0.2753
Zelkowitz	0.2145	0.2366	0.1806	0.2990
Brooks	0.2592	0.2922	0.2276	0.3536
Ambler	0.3082	0.2775	0.3415	0.2010
Benington	0.3850	0.3197	0.3750	0.4202
Boehm	0.3906	0.2839	0.3960	0.3818

In the fourth filter, Table 4.4 shows the vector magnitudes of the RoT compared to the mean vectors for either Traditional or Agile processes. Yang is the best fit for Traditional methods, the preponderance of the data. Papatheocharous is handily the best fit for Agile development, with the smallest LSM value in any test ran on any category. It is interesting to note that both Yang and Papatheocharous better predicted Agile than Traditional when the table is reviewed horizontally, a trend that does not hold for most other RoT in the table.

Table 4.4: LSM Magnitude by Development Process

Rule of Thumb	**Traditional**	**Agile**
N	*932*	*186*
Yang	**0.102**	0.0678
Borysowich	0.1273	0.147
Papatheocharous	0.1556	**0.0141**
Sommerville	0.1849	0.2977
Heijstek	0.2214	0.0548
Thibodeau	0.2005	0.2786
Zelkowitz	0.198	0.3385
Brooks	0.2454	0.3608
Ambler	0.3197	0.2421
Benington	0.3766	0.4186
Boehm	0.3876	0.3750

In the final LSM table, 4.5, we compared the RoT fit against six categories of size. Results are consistent with previous tests, with Yang fitting the best overall in five of 6 size categories. Papatheocharous and Borysowich were second and third best fit. For projects larger than 250K ESLOC, the Papatheocharous rule fit best.

Table 4.5: LSM Magnitude by Size

Rule of Thumb	**<5K**	**5-20K**	**20-50K**	**50-100K**	**100-250K**	**>250K**
N	*104*	*188*	*174*	*113*	*105*	*38*
Yang	**0.1105**	**0.1049**	**0.1225**	**0.0316**	**0.0990**	0.1342
Borysowich	0.1625	0.1400	0.1517	0.1030	0.1049	0.2437
Papatheocharous	0.1631	0.1581	0.1772	0.0849	0.1517	**0.1049**
Sommerville	0.2177	0.1965	0.1897	0.2205	0.1673	0.3813
Heijstek	0.2267	0.2249	0.2425	0.1490	0.2126	0.1208
Thibodeau	0.2417	0.2140	0.2135	0.2205	0.1766	0.3813
Zelkowitz	0.2223	0.2064	0.1924	0.2502	0.1871	0.4104
Brooks	0.2581	0.2550	0.2354	0.2839	0.2223	0.4445
Ambler	0.3455	0.3140	0.3479	0.2891	0.3450	0.1881
Benington	0.4243	0.3906	0.3960	0.3844	0.3544	0.5324
Boehm	0.4465	0.4010	0.4202	0.3750	0.3718	0.4893

4.2.2 Hotelling's Test

While the LSM method in the previous section is descriptive, its statistical significance is not robust. Because of the weak findings, we turned to Hotelling's T^2 Test to add statistical rigor and act as a check on the LSM results. The Hotelling Test measures the vector distribution of data against the a proposed set of means, in this case the existing RoT. The null value assumes the means are equivalent, which is the desired result for at least one of the RoT in each data filter. The alpha value was set at $\alpha = 0.10$. Select T^2 values and p-values are given in Table 4.6 below.

Of all tests run, filtering the data each and every way, every single test resulted in accepting the null hypothesis. This inconclusive finding states that every RoT described in this thesis is statistically applicable to every slice of the data. The most likely reason is the wide variance in the data. While specific projects may reject the null for certain RoT, the uncertainty band at each phase from the sum of projects is so wide it effectively encompasses each RoT. This variance is pervasive regardless of categorical filter applied to the data. Because of this variance, we see that nearly every rule is 'statistically' predictive' of our data. When every rule is predictive, none is predictive. DoD performance is simply either too diverse or too erratic to consistently fit a single rule or benefit from a rule.

The lack of statistical significance of the Hotelling's Test findings goes against what the eyeball test and the LSM test tell us. But one should keep in mind the objective—to help establish some heuristic, some rough guideline. And heuristics implicitly recognize inadequacy and lack of uniformity. The LMS test seems to show that, when asked *"Which rule is marginally superior among several equivalent rules?"*, we can reply that Yang, Borysowich, and Papatheocharous are good places to start. Now that we understand the general phase distribution pattern and have attempted to identify best-fit rules of thumb, the remainder of the chapter is devoted to analysis

of the relationships between phases.

Table 4.6: Select Hotelling's Test Results

Rule of Thumb	**Sample**	$\mathbf{T^2}$	**p-value**
Yang	Full dataset	0.27024	0.9917
Yang	Service = Navy	0.24719	0.993
Yang	Super Domain = AIS	0.67141	0.9576
Yang	Super Domain = RT	0.15537	0.9971
Yang	Process = Agile	0.49275	0.975
Benington	Full dataset	3.227	0.5224
Benington	Service = Navy	2.8122	0.5937
Benington	Super Domain = AIS	4.3008	0.4002
Benington	Super Domain = RT	2.4235	0.6602
Benington	Process = Agile	5.1801	0.2879
Ambler	Full dataset	1.5284	0.8223
Ambler	Service = Navy	1.4953	0.8292
Ambler	Super Domain = AIS	0.64385	0.9606
Ambler	Super Domain = RT	1.5693	0.8153
Ambler	Process = Agile	1.5794	0.8239
Papatheocharous	Full dataset	0.67978	0.954
Papatheocharous	Service = Navy	0.70573	0.9511
Papatheocharous	Super Domain = AIS	1.4817	0.8404
Papatheocharous	Super Domain = RT	0.58622	0.9648
Papatheocharous	Process = Agile	1.1443	0.891
Borysowich	Full dataset	0.66608	0.9556
Borysowich	Service = Navy	0.61852	0.9614
Borysowich	Super Domain = AIS	1.1495	0.8934
Borysowich	Super Domain = RT	0.58685	0.9647
Borysowich	Process = Agile	1.404	0.8487

4.3 Overall Distribution Patterns

The previous section compared the final phase distributions to the rules of thumb to determine their validity. This section looks at the general distribution of effort in the database and analyzes the change in effort from initial to final, looking to see if there's a noticeable change in phasing distribution. Figure 4.3 shows one way to view the results. It traces the mean percent effort for each phase and compares the initial distribution to final.

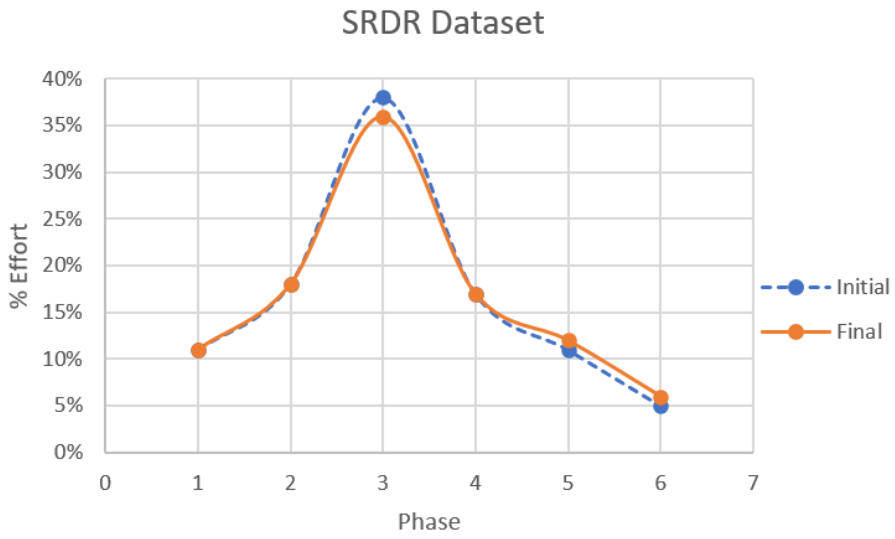


Figure 4.3: Mean Percent Effort Distribution Overall, by Phase

The figure shows that the percentage of effort *does not* change much between the initial estimate and final result. Whatever occurs during the development process, the proportional effort allocation remains static overall. The average change of the proportion of effort from initial estimate to final result is presented in Table 4.7. It also shows these changes for multiple categorical filters on the data. Each column presents the percent change in effort for each phase from initial to final, calculated by taking the differential for each project and then averaging the results. A positive figure represents an increase in effort from initial to final, while a negative figure

represents a decrease in effort, for that particular phase. Except for rounding, each row balances to zero since a percent increase in one phase requires a percent decrease in another.

Most results show limited change in the percentage of each phase of the project. Looking at the top row for the full dataset, there is no more than a two percent change in effort from initial to final for any phase. Broken out into different categories of programs, the differences also look minor. The largest differences are to large projects above 250K ESLOC, which showed a 9% and 6% change in the first two phases. There were some large changes to Space and Sea Operating Environments as well as the two Super Domains of MS and AIS, which showed multiple phase changes between 5 and 7 percent. As suspected based on the variance in the data, when these large-change results were tested for significance between the initial and final effort percent allocation, no significance was found in any of them.

The similarity in proportional effort described should not be taken to assume the DoD often gets the actual effort estimates correct for each phase, since the majority of projects experienced a large amount of growth in total effort hours. The mean and median growth in size and effort for the database is shown in Table 4.8. Size is related to manhours, as a larger project takes more time to code and test. The table shows that both size and effort increase, with a slightly greater percent change in size than effort. This indicates our problems may exist beyond correctly proportioning effort by phase.

Table 4.7: Difference in Means of Proportional Effort for Each Phase, Initial to Final

	1	2	3	4	5	6
Overall	0%	1%	1%	0%	-2%	-1%
Service						
Air Force	-3%	0%	-1%	0%	2%	2%
Army	1%	0%	0%	-1%	2%	-2%
Navy	1%	-4%	-2%	2%	1%	2%
Super Domain						
AIS	-3%	-4%	**6%**	-2%	-1%	3%
ENG	-1%	-1%	-3%	3%	1%	2%
MS	-2%	**-5%**	1%	-2%	**5%**	3%
RT	0%	-2%	-1%	0%	2%	1%
Operating Env.						
Air	-3%	-2%	-1%	2%	2%	2%
Sea	**6%**	-3%	**-6%**	2%	0%	1%
Surface	1%	-1%	1%	-2%	2%	-1%
Space	-1%	-2%	-3%	-3%	2%	**7%**
Process						
Agile	0%	-1%	-1%	0%	2%	1%
Traditional	-1%	-3%	-1%	3%	1%	0%
Initial Size						
<5K	-2%	1%	-3%	1%	3%	0%
5-20K	-1%	-2%	0%	0%	2%	1%
20-50K	-1%	-2%	0%	1%	1%	1%
50-100K	3%	-3%	-3%	1%	1%	0%
100-250K	0%	-3%	1%	-1%	1%	2%
>250K	**9%**	**-6%**	-2%	1%	-3%	1%

Table 4.8: Percent Increase in Size and Effort

		%Increase ESLOC		*%Increase Effort*	
	N	Mean	Median	Mean	Median
Full Dataset	309	136%	37%	113%	33%

As an extension of these results, we plotted proportional effort against total effort hours to understand the disconnect between proportional effort and actual effort. The mean change in total effort hours is presented in the top of Figure 4.4, with the bottom a repeat of Figure 4.3. The stack of both figures allows us to compare the differences between total and proportional effort change for each phase. While the proportion of effort does not change between initial and final, it appears that the total effort *does* increase. It is interesting that phase 3 effort does not noticeably increase, but the first two and last three phases do. Reflecting on the literature, increases in early effort may cause increases in the last phases, but not necessarily phase 3. Actual mean values for each phase, by category, are presented in the Appendix for further review.

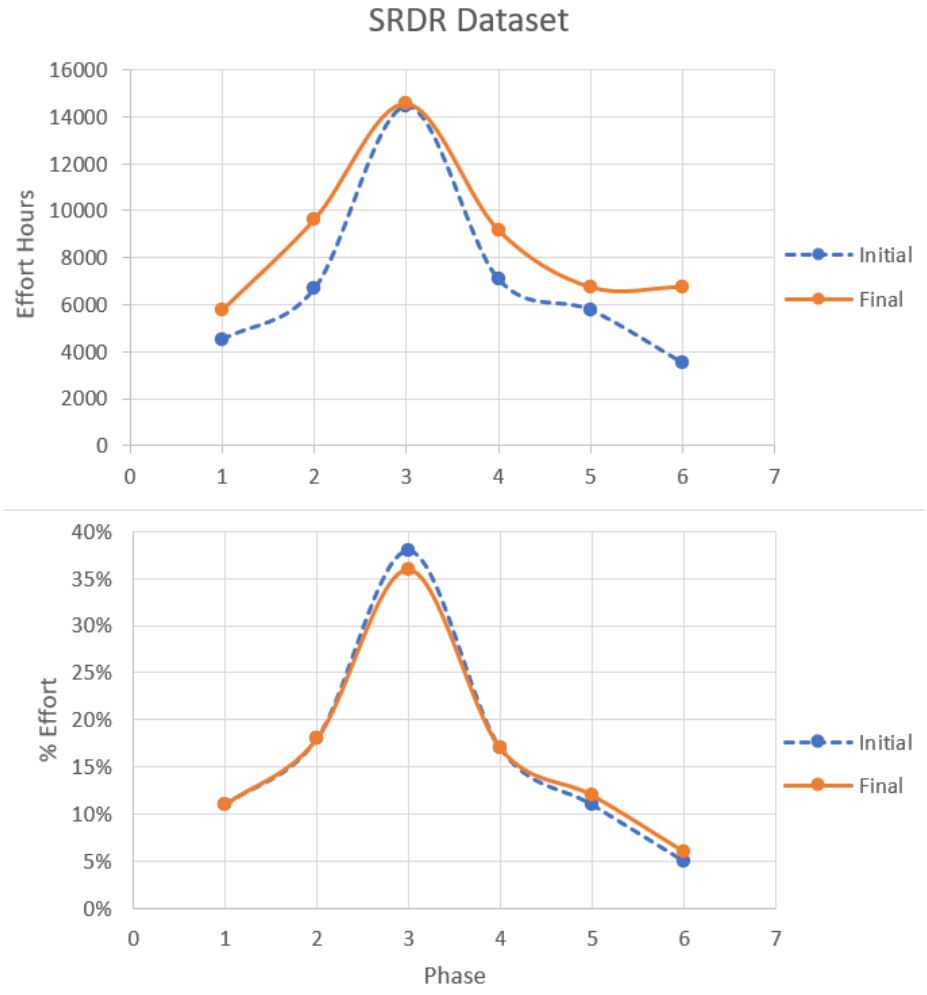


Figure 4.4: Mean Effort Hours (top) and Mean Percent Effort Distribution (bottom) Overall, by Phase

It is important to note that this analysis provides no commentary on whether estimates were accurate, either in cost or schedule. It only shows that the initial *proportion* of effort of each phase was quite accurate, on *average*, compared to the proportion of effort at the end, even though the *total* effort and size grew. One might be inclined to conclude, then, that the phases themselves do not play a large part in DoD's management problems.

To evaluate our estimation accuracy, we calculated the percent error in each phase from initial to final. We did this by subtracting initial effort hours from final effort hours and dividing by the initial effort hours. A positive value/error indicates the effort grew, while a negative value/error indicates the effort decreased. A graph of both the mean and median percent error for our dataset is provided in Figure 4.5. The left axis corresponds to the median values, while the much larger right axis corresponds to the mean error values. The smallest errors are found in the first two phases, with a minor peak in phase 3 and a decrease (compared to phase 3 error). Phase 6 experiences the highest error overall, likely due to inconsistency in phase 6 reporting and the smaller average value. While the results are not statistically significant in their own right, they do indicate that our effort allocation in the first two phases is the most accurate, and the effort allotted to phases 3 and 6 are the least accurate. This may help the program manager focus risk analysis to the least accurate phases.

Going a step further, we regressed the percent change in schedule on the percent error in each phase. The regression output is in Figure 4.6. While the percent error values are not all explanatory, the first two phase errors are significant. This is a critical finding; that early phase error is the most explanatory to schedule growth compared to the later phases, even though Figure 4.5 showed that the actual percent error in the early phases were the smallest. So, while we are better at estimating our

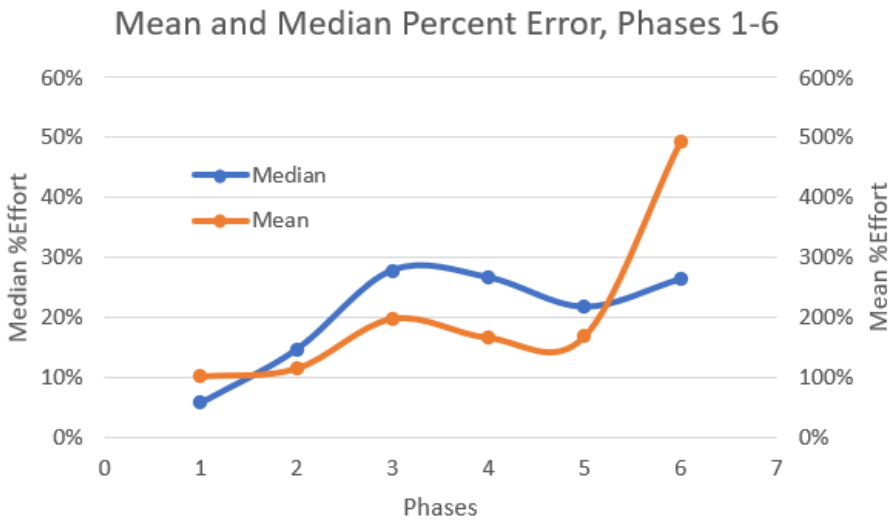


Figure 4.5: Estimation Error by Phase

effort in early phases, the estimate itself is significant in explaining future schedule growth.

We also evaluated the correlation between total schedule growth and total effort. The two factors are positively correlated, with a coefficient of 0.3484 and an associated p-value of <0.0001. Full correlation test results are in the appendix. The results indicate that schedule increases tend to accompany increases in total effort on a project, which ties in to Thibodeau's research that showed schedule slip necessarily causes more effort to be allotted.

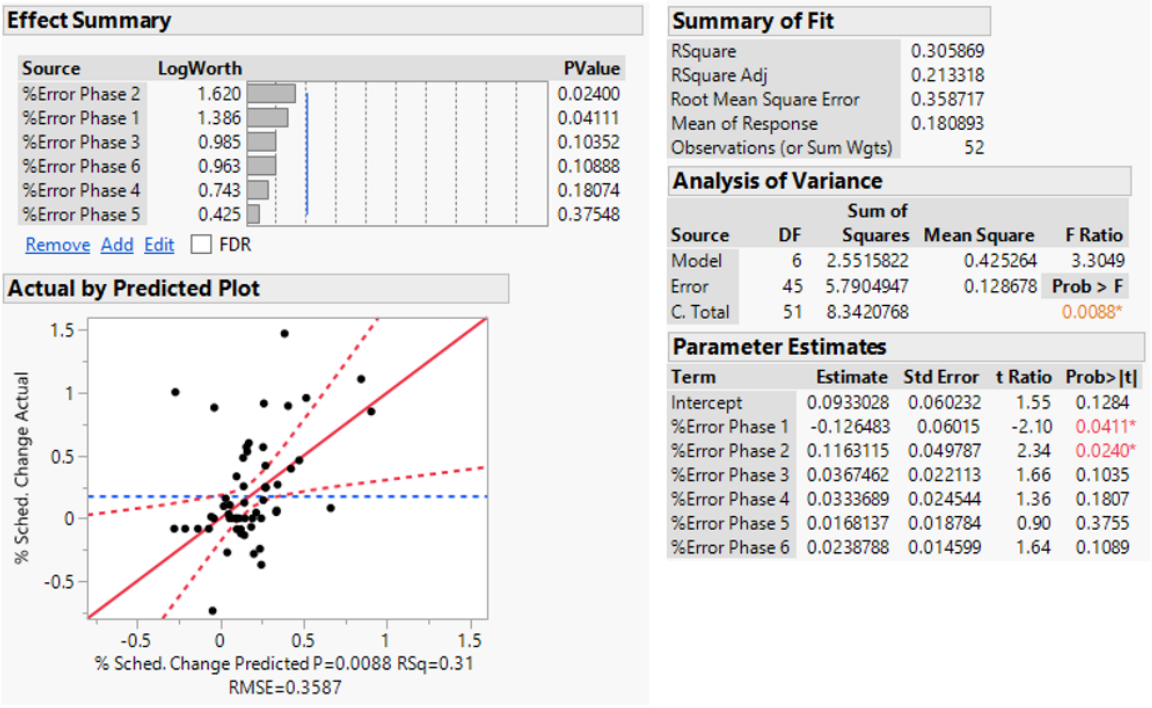


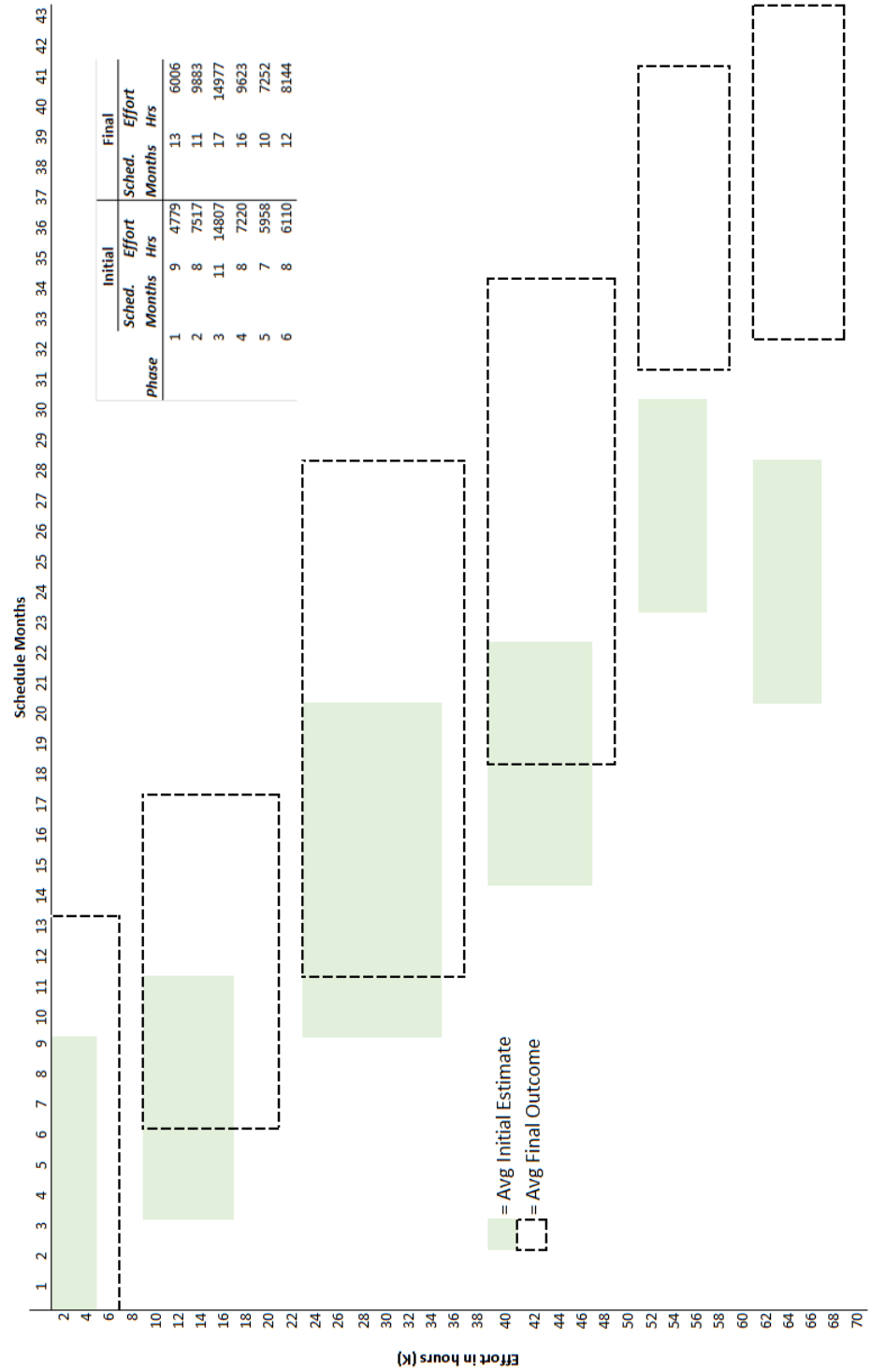
Figure 4.6: Regression of Percent Schedule Growth by Percent Error, all Phases

4.3.1 Mean Schedule and Effort Gantt Analysis

To emphasize this point and study the patterns in Figure 4.4, we calculated the average schedule months and effort for each phase, initial and final, using numbers from earlier tests. We then plotted the values to recreate Thibodeau's Gantt-like chart from 1980 using the current SRDR data. The result is in Figure 4.7. The solid green boxes are the initial estimate, and the dashed boxes are the final outcome. Each box pair corresponds to an SRDR phase, with Phase 1 in the top-left and Phase 6 in the bottom-right. The horizontal axis is schedule (time), and the vertical axis is effort hours measured in thousands. The chart reflects a similar trend to Thibodeau's chart. One not only sees growth in effort and schedule for each phase, but can also track schedule slips across time.

Proportional growth factors for schedule and effort are presented in Table 4.9, allowing one to see the relationship of schedule growth to effort growth. The Schedule and Effort factors are calculated by dividing the final average schedule or effort by the initial value. The last two growth factors are calculated by dividing the effort growth factor by that of schedule (and vice-versa), and function somewhat like a comparative elasticity. The Effort/Schedule factor represents how much Effort grows in relation to growth in Schedule. The Schedule/Effort factor represents the opposite, how much Schedule grows in relation to growth in Effort. Numbers close to 1 mean that schedule and effort grow in tandem, while numbers higher/lower than 1 indicate effort growing more/less than comparable growth in schedule. For phase 1, the E/S factor *0.88* can be interpreted as effort growing 12% less than a unit increase in schedule. The same row's S/E factor of *1.14* means that schedule increases 14% more than effort increases in phase 1.

All growth factors for the schedule and effort columns are greater than 1, indicating that schedule and effort increase regardless of phase. As an important complement to



Phase	Initial Sched. Months	Initial Effort Hrs	Final Sched. Months	Final Effort Hrs
1	9	4779	13	6006
2	8	7517	11	9883
3	11	14807	17	14977
4	8	7220	16	9623
5	7	5958	10	7252
6	8	6110	12	8144

Figure 4.7: Gantt Chart of Average Effort versus Schedule in each phase, Comparing Initial to Final

Table 4.9: Growth Factors for Change in Schedule Months and Total Effort, by Phase

Phase	Schedule	Effort	Effort/Schedule	Schedule/Effort
1	1.43	1.26	0.88	1.14
2	1.39	1.31	0.95	1.06
3	1.54	1.01	0.66	1.52
4	1.94	1.33	0.69	1.45
5	1.57	1.22	0.77	1.29
6	1.53	1.33	0.87	1.15
Overall	1.57	1.24	0.80	1.27

Figure 4.4, we see significant growth in both cost and schedule. The schedule growth factors range between 1.39 and 1.94, while the effort growth factors range from 1.01 to 1.33. The overall growth factor for schedule was 1.57, with phase 2 growing the least and phase 4 growing the most. The growth in effort followed a different pattern; average growth was a factor of 1.24, with the lowest growth in phase 3 and phases 2, 4, and 6 roughly tying for highest growth. There was no proportional schedule growth for phase 3, but there was considerable overall schedule growth in that phase.

Looking at the last two columns a few key observations are apparent. The relative elasticity between schedule and effort is highest in phase 2, where the change in effort and change in schedule increase proportionally. The correlation is weakest in phase 3, where an increase in schedule by 1 unit only increases relative effort by 0.66 (phase 4 is close at 0.69). Viewed another way using the S/E factor of 1.52, any increase in effort will cause a much larger increase in schedule by 52%. These factors illustrate the idea that schedule extensions or delays in phase 2 have a greater effect on required effort than during phase 3 (or 4 for that matter), and likely cause the disparity seen in phase 3. This will be explored further in the next section.

4.4 Phase Relationship Analysis

This section studies if, and how, early effort allocation relates to schedule growth. The histogram in Figure 4.8 shows the degree of schedule growth for the whole dataset of 309 projects for which we had initial and final SRDR submissions. Growth is conveyed as a factor, with "0" meaning no schedule increase and "1" meaning a 100 percent increase (or doubling of schedule). Negative values reflect a shorter schedule than estimated.

The median schedule increase was 17 percent and, given the right skew in the data, the mean schedule increase was 52 percent. This put the mean at the 70th percentile. There was a significant range of schedule change. Six projects had over 400 percent increase, while 97 projects (31% of total) had no growth or experienced a schedule decrease.

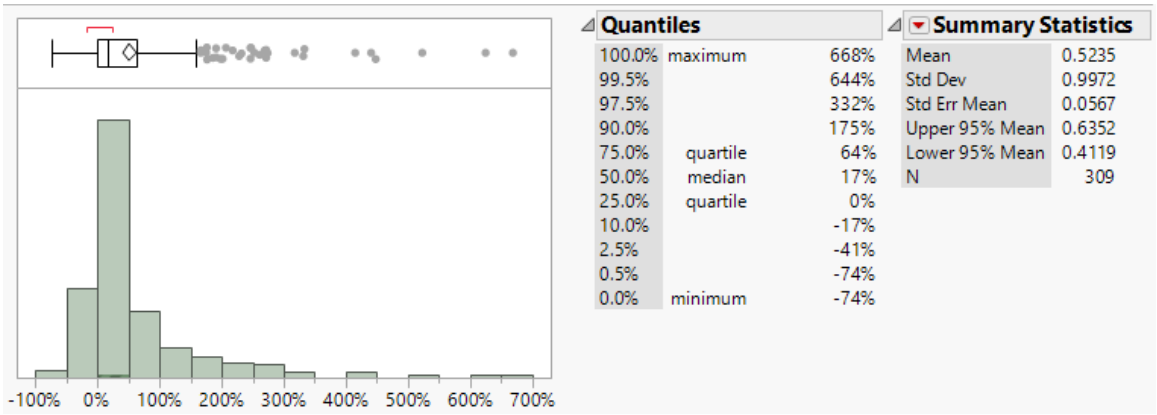


Figure 4.8: Histogram and Summary Statistics of Percent Schedule Increase

Given the wide range of schedule growth, several questions arise. First, are there different phasing distributions among those projects with higher-than-average schedule growth, compared to projects with lower-than-average growth? Is there a different phasing distribution for projects that experienced negative schedule growth (a decrease in expected time to complete)? Are results the same when using the median

growth instead of mean growth?

In this first section, we examined the change in *scheduled months* for the entire project. We wanted to see how the final effort distribution differed between projects that experienced high versus low schedule growth. To do this, we split the data into two cohorts; those that experienced a larger than average schedule growth (greater than 52 percent) and those that experienced less than average growth. We then compared the phase distributions between the two groups. The same analysis was also performed using the median schedule growth (17 percent increase) as the split between cohorts. Given the emphasis in the literature that early phases are critical to success, we look at how the first three phases may have differed between these two cohorts. The test sequence "B" was selected for this analysis to simplify phase definitions.

A	1	2	3	4	5 + 6
B	1	2	3	4 + 5 + 6	
C	1 + 2		3	4 + 5 + 6	

The mean split phase values are in Figure 4.4. As anticipated by the literature, the projects that had a greater percentage of effort in Phase 1 and 2 experienced a smaller (or negative) total schedule growth. Projects with below-average growth reported 30 percent of effort in the first two phases, compared to 21 percent for the projects with above-average growth. This is a 9 percent difference in effort. To note, the effort difference for Phase 3 was comparatively small, which possibly indicated that schedule growth issues do not typically arise in this phase.

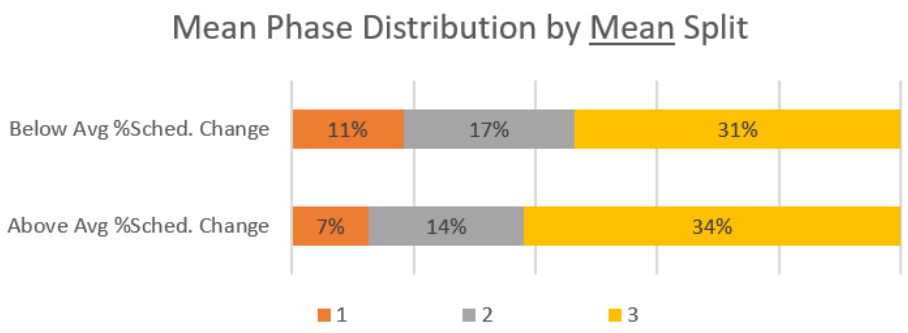


Figure 4.9: Mean Phase Distribution by Mean Split

Splitting by the median, in Figure 4.4, resembles the Mean split results, with projects that invested a greater percentage of effort in Phase 1 and 2 experiencing below-median schedule growth. Projects with below-median growth reported 28 percent of effort in the first two phases, compared to 22 percent for the projects with above-median growth, a 6 percent difference. The effort for Phase 3 was unchanged in the two groups, lending credence to our hypothesis that the first two phases are critical to preventing schedule growth.

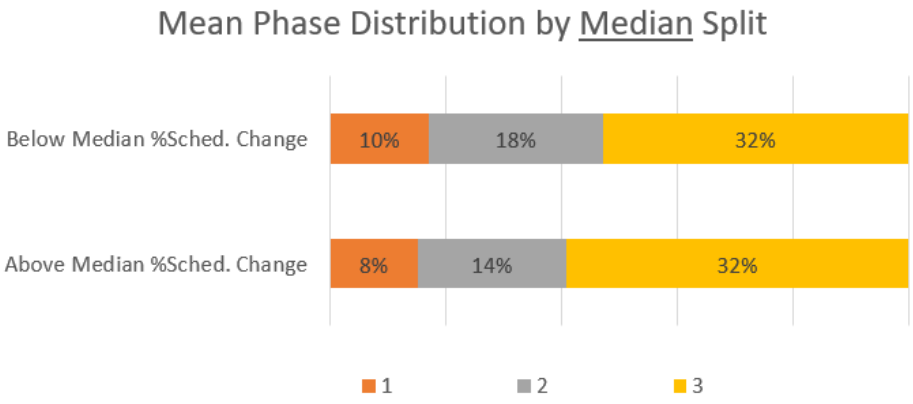


Figure 4.10: Median Phase Distribution by Median Split

Given the statistical association in the literature between the first two phases and schedule growth, we ran a series of t-tests on multiple categories of data (Boehm,

1987; Brooks, 1975; Thibodeau and Dodson, 1980). The analysis compared the effort for projects with lower-than-average schedule growth to the effort for projects with higher-than-average schedule growth. This analysis is shown in Table 4.10, with the average growth pegged at 52 percent increase in schedule. This put the mean growth in the 70th percentile. Highlighted values in the p-value column indicate where the difference between the below-mean and above mean cohorts is statistically significant.

Table 4.10: Difference in Phase 1+2 Percent Effort, Above and Below Mean of 52% Growth

Comparison of Avg. Percent Effort in Phase 1+2 Between the Above-Mean Cohort and the Below-Mean Cohort

	N	Below Mean of 52%	Above Mean of 52%	Delta	t Ratio	p-value
Overall	308	30%	21%	9%	-3.966	0.0001
Service						
Air Force	76	28%	20%	8%	-2.129	0.0366
Army	96	29%	17%	12%	-1.775	0.0791
Navy	137	32%	23%	9%	-3.437	0.0008
Super Domain						
AIS	25	36%	21%	15%	-2.249	0.0344
ENG	39	29%	18%	11%	-2.155	0.0377
MS	12	36%	31%	5%	-0.435	0.6726
RT	233	29%	21%	8%	-3.027	0.0027
Application Domain						
C&C	66	26%	21%	5%	-1.119	0.2711
CAS	11	42%	19%	23%	-3.541	0.0489
COM	22	38%	31%	7%	-0.375	0.4664
M&F	1	.	33%			
MP	14	30%	23%	7%	-1.158	0.273
RTE	54	30%	20%	10%	-2.307	0.0250
S&S	12	28%	17%	11%	-1.197	0.2588
SP	35	28%	29%	-1%	0.219	0.828
SS	20	31%	12%	19%	-2.163	0.0442
TMDE	7	25%	26%	-1%	0.116	0.912
TOOL	6	35%	33%	2%	-0.086	0.9352
TRN	6	39%	31%	8%	-0.463	0.667
VC	33	29%	11%	18%	-2.451	0.0201
VP	22	24%	12%	12%	-1.698	0.105
Operating Env.						
Air	137	25%	22%	4%	-1.273	0.2052
Sea	32	30%	18%	12%	-1.401	0.1715
Space	15	19%	10%	9%	-1.620	0.1286
Surface	125	34%	25%	9%	-2.580	0.0110
Process						
Agile	206	30%	20%	10%	-3.914	0.0001
Traditional	103	29%	25%	4%	-1.225	0.2235
Initial Size						
<5K	52	26%	14%	12%	-1.678	0.0995
5-20K	97	31%	21%	10%	-2.952	0.004
20-50K	80	28%	21%	7%	-2.177	0.0325
50-100K	40	33%	25%	8%	-1.116	0.2716
100-250K	28	30%	29%	1%	-0.197	0.8457
>250K	12	29%	25%	4%	-0.296	0.7734

The overall dataset showed a significant relationship between the sum of effort conducted in the first two phases and whether the program will experience greater than or less than average schedule growth. More effort is generally associated with below average schedule growth, though the association was mixed depending on the various categories of analysis. For the overall dataset and each Service, though, the association held. The results by category are discussed below.

All Super Domains except MS were significant. MS consists of the TOOL and TRN domains, neither of which were significant individually. The Application Domains were a mixed bag, with about a third of the tests significant. All ADs that tested significant had a large % Delta value; the CAS domain below/above mean values were 42%/19%, with a massive percent change of 23%. Only one Operating Environment was significant, Surface. Though not significant, the percent change for the Sea and Space OEs were quite large. Among the two Process types, Agile proved very significant. Traditional development did not show significance.

Among the six categories for size, the smallest three size bins were significant, while the largest three size bins were not significant. The finding suggests that the relationship between large upfront effort and lesser schedule growth is only true for small programs.

There are a few considerations while interpreting Table 4.10. First, the table is only intended to be read horizontally; one cannot infer vertical differences in means. That is, AIS percent effort at 36% may or may not be statistically different than ENG's 29%. It can only be claimed that the percent effort distribution for AIS is significantly different between the above-mean cohort and below-mean cohort for AIS alone. The breakout of the table has two aims. First, it delves below the general findings, and shows where the findings are statistically significant. Those are tested horizontally. Second, it aims to assist the practitioner looking for a more specific

heuristic for what the mean values may be for the phases of programs with various characteristics. Vertically, those means were not tested to be distinct from other members in each category, thus they can only be notional values, and not necessarily superior to using the overall values of 30% and 21% as a rule of thumb.

We repeated the above analysis for a median divide, shown in Table 4.11. The findings from the mean divide persisted. The overall dataset phase effort difference between the median cohorts decreased from 9% to 7% but maintained significance. All significant tests from the mean cohorts were the same; all Services were significant, 3 of 4 SDs, four ADs, and the smallest three Size bins were significant. One additional category was significant—the Traditional development process. This means that with a median split, both categories of development process are significant, not just Agile as with the mean split.

Table 4.11: Difference in Phase 1+2 Percent Effort, Above and Below Median of 17% Growth

Comparison of Avg. Percent Effort in Phase 1+2 Between
the Above-Median Cohort and the Below-Median Cohort

	N	Below Median of 17%	Above Median of 17%	Delta	t Ratio	p-value
Overall	308	31%	24%	7%	-3.711	**0.0002**
Service						
Air Force	76	29%	20%	9%	-2.508	**0.0143**
Army	96	30%	22%	8%	-1.882	**0.0623**
Navy	137	33%	26%	7%	-2.648	**0.0091**
Super Domain						
AIS	25	37%	25%	12%	-1.933	**0.0655**
ENG	39	33%	18%	15%	-3.321	**0.002**
MS	12	25%	40%	-15%	1.461	0.1747
RT	233	30%	23%	7%	-2.796	**0.0056**
Application Domain						
C&C	66	31%	19%	12%	-2.606	**0.0114**
CAS	11	43%	25%	18%	-2.079	**0.0673**
COM	22	38%	33%	5%	-0.54	0.595
M&F	*1*					
MP	14	30%	25%	5%	-0.616	0.5492
RTE	54	29%	24%	5%	-1.239	0.2211
S&S	12	31%	21%	10%	-1.221	0.25
SP	35	24%	32%	-8%	1.481	0.1482
SS	20	37%	13%	24%	-3.834	**0.0012**
TMDE	7	25%	26%	-1%	0.1162	0.912
TOOL	6	25%	53%	-28%	2.116	0.1018
TRN	6	25%	35%	-10%	0.4369	0.6847
VC	33	34%	19%	15%	-2.619	**0.0135**
VP	22	23%	18%	5%	-0.668	0.5116
Operating Env.						
Air	137	28%	22%	6%	-2.128	**0.0352**
Sea	32	32%	24%	8%	-0.973	0.3385
Space	15	19%	10%	9%	-1.623	0.1286
Surface	125	34%	29%	5%	-1.503	0.1353
Process						
Agile	206	30%	23%	7%	-2.853	**0.0048**
Traditional	103	32%	24%	8%	-2.492	**0.0143**
Initial Size						
<5K	52	30%	16%	14%	-2.441	**0.0078**
5-20K	97	31%	24%	7%	-0.018	**0.0766**
20-50K	80	29%	23%	6%	-1.841	**0.0695**
50-100K	40	35%	25%	10%	-1.597	0.1186
100-250K	28	30%	30%	0%	-0.133	0.8955
>250K	12	24%	32%	-8%	0.668	0.5188

We then focused our cohort analysis on comparing those programs which experience schedule change in the upper quartile (measured as $> 64\%$ growth) versus those that experienced schedule change in the lower quartile (measured as $\leq 0\%$ growth, or a reduction in schedule). (See Table 4.12.) This last approach analyzes how the best and worst performers on a schedule growth basis compared in their effort allocations. The results were consistent, and interestingly, the relationship showed no strengthening of effect. For the mean divide, the efforts were 30 and 21 percent, and for the extreme quartiles, the efforts were 29 and 20 percent.

Table 4.12: Difference in Phase 1+2 Percent Effort, Top and Bottom Quartiles

Comparison of Avg. Percent Effort in Phase 1+2 Between
the Top and Bottom Schedule Growth Quartiles

	N	Bottom Quartile	Top Quartile	Delta	t Ratio	p-value
Quartile Extremes						
	BQ=97; TQ=77	29%	20%	9%	-4.945	0.0001

As a further test of difference between schedule cohorts, we compared the phase allocation for the cohort that experienced zero or negative schedule growth against the cohort that did experience schedule growth. The results are shown in Table 4.13. To note: the first cohort definition nearly matches that of the 25th percentile from the prior test, but the projects between the two cohorts are not entirely the same. The table shows that the delta is smaller than in Table 4.4, but the difference is still evident. There is a significant difference between the effort allocations of the projects that don't experience schedule growth and those that do.

Table 4.13: Difference in Phase 1+2 Percent Effort, Growth vs. Zero/Neg Growth

Comparison of Avg. Percent Effort in Phase 1+2 Between
Zero or Negative Schedule Growth and All Other Growth Cohorts

	N	%Growth≤ 0	%Growth>0	Delta	t Ratio	p-value
Overall	309	32%	25%	7%	3.23	0.0001

4.4.1 Marginal Contingency Analysis - Odds

To delve deeper in our analysis, we calculated the odds ratios for schedule growth given decreasing values of Phase 1+2 percent effort. Using contingency tables and the deciles of phase 1+2 percent effort, we evaluated the marginal change in the odds of experiencing schedule growth (where %Sched. change > 0). The results are in Table 4.14. The top row, 10/90, indicates that 10 percent of the data has a phase 1+2 percent effort higher than 50%, and 90 percent of the data has a smaller phase 1+2 percent effort. The last column for this row indicates that the odds of experiencing schedule growth are 1.55 times higher if you allot less than 50% effort in phase 1+2, compared to allotting more than 50% effort. A graph of the odds ratios vs percent effort is in Figure 4.11.

Table 4.14: Marginal Contingency Analysis - Odds of Schedule Growth for Different Cohorts of Phase 1+2 %Effort

Quantiles, Above/Below Phase split value	**%Phase 1+2 split value**	**Odds of growth if smaller Phase1+2 allocation used**
10/90	50%	1.55
20/80	43%	2.67
30/70	36%	2.01
40/60	33%	1.71
50/50	27%	1.64
60/40	23%	2.18
70/30	17%	2.91
80/20	10%	2.85
85/15	5%	3.33
90/10	2%	4.02

The odds ratios are somewhat scattered as one increases early percent effort, but do fit an overall negative trend. As the phase 1+2 split increases from 2% to 50%, the odds ratio decreases. By allocating more and more effort to the first two phases, the odds of experiencing schedule growth generally decrease. The fact that all odds ratios are greater than 1, regardless of early phase allocation, points to two findings:

one, that schedule growth is relatively common in the dataset, and two, that since we know some projects don't experience schedule growth, the early phases are not the final decision on whether or not schedule growth occurs at all. This will be further examined in the following robustness checks section.

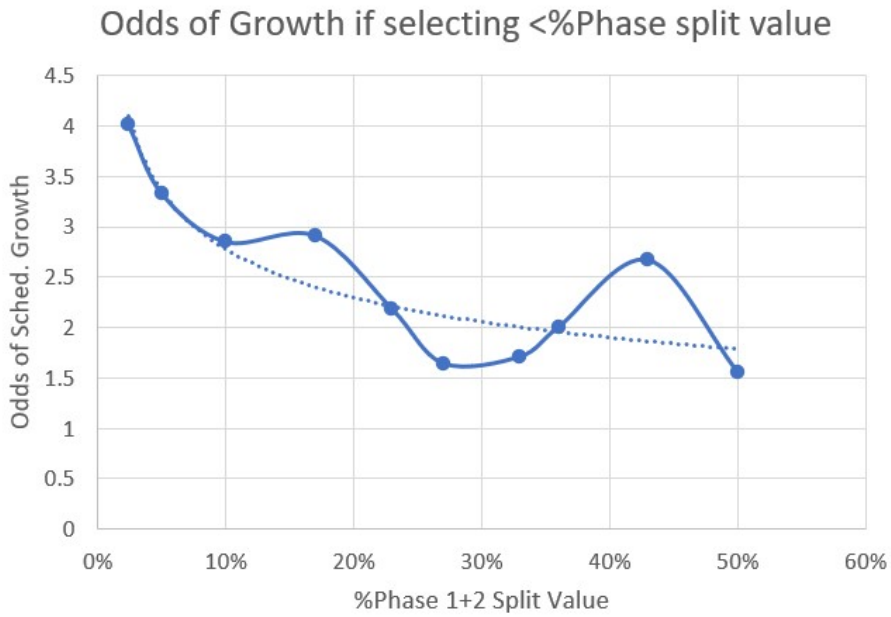


Figure 4.11: Odds Ratio of Schedule Growth Given Percent Effort in Phase 1+2

4.5 Robustness Checks

4.5.1 Robustness Checks on Phase Distributions

Our phase distribution data was a swiss-cheese of sorts, with multiple reports omitting phase effort for one or more phases. Of the 1128 reports studied, just 132 reports included effort hours for each of the six phases. We wanted to see if the smaller group of "complete" data that did include values for all phases had a different distribution of effort than the larger dataset studied. The mean and median percent phase values for the 'complete" dataset are shown in Figure 4.12. Encouragingly, the results are not far off from the larger dataset. The primary shift in effort appears to be from Phase 3 to Phases 5 and 6. This checks out, since we needed to combine phases 5 and 6 for our tests due to the variation in effort documentation; omitting reports that allocate zero hours to either phase necessarily increases the mean values. A few of the application domains show larger changes in phase 1 and 2 effort, but we can ignore these effects since application domain is not significant in our overall research.

		Final Percent Effort, Mean Values						Final Percent Effort, Median Values					
	N	Phase 1	Phase 2	Phase 3	Phase 4	Phase 5	Phase 6	Phase 1	Phase 2	Phase 3	Phase 4	Phase 5	Phase 6
Full Dataset	132	11%	18%	30%	16%	10%	15%	8%	16%	27%	13%	8%	10%
Service													
Air Force	29	10%	16%	33%	15%	11%	16%	8%	13%	30%	8%	7%	16%
Army	19	10%	24%	32%	18%	7%	9%	8%	26%	30%	15%	4%	4%
Navy	84	11%	17%	29%	16%	11%	16%	8%	16%	26%	16%	9%	11%
Super Domain													
AIS	11	11%	20%	43%	8%	6%	12%	9%	16%	40%	8%	2%	10%
ENG	22	9%	18%	32%	13%	9%	19%	7%	17%	33%	10%	6%	13%
MS	5	12%	14%	35%	18%	10%	12%	11%	15%	44%	18%	12%	9%
RT	94	11%	18%	28%	18%	11%	15%	7%	20%	26%	17%	9%	10%
Application Domain													
C&C	21	11%	17%	37%	16%	8%	12%	8%	20%	30%	13%	6%	8%
CAS	4	12%	29%	39%	6%	6%	9%	15%	35%	40%	4%	2%	4%
COM	15	7%	27%	30%	16%	10%	10%	6%	29%	29%	17%	6%	9%
M&F	0	-	-	-	-	-	-	-	-	-	-	-	-
MP	7	11%	14%	46%	9%	6%	14%	9%	12%	40%	8%	2%	12%
PC	1	7%	18%	60%	5%	4%	6%	7%	18%	60%	5%	4%	6%
RTE	22	9%	15%	23%	23%	14%	16%	7%	13%	22%	19%	11%	13%
S&S	6	11%	17%	28%	14%	10%	21%	12%	19%	27%	11%	9%	10%
SP	12	9%	18%	27%	18%	6%	22%	5%	21%	25%	15%	5%	24%
SS	10	7%	14%	28%	16%	13%	22%	5%	12%	34%	11%	9%	24%
TMDE	5	8%	27%	42%	8%	2%	13%	5%	20%	40%	3%	1%	6%
TOOL	4	10%	15%	32%	21%	9%	13%	11%	15%	32%	20%	8%	9%
TRN	1	17%	8%	47%	4%	16%	8%	17%	8%	47%	4%	16%	8%
VC	13	21%	15%	21%	12%	14%	18%	18%	11%	19%	10%	13%	10%
VP	11	11%	16%	27%	18%	16%	12%	9%	21%	26%	18%	14%	6%
OE - Summary													
Air Vehicle	56	11%	13%	31%	17%	11%	17%	8%	12%	27%	16%	9%	13%
Sea System	0	-	-	-	-	-	-	-	-	-	-	-	-
Space System	6	8%	15%	39%	25%	6%	7%	6%	12%	41%	16%	3%	6%
Surface	70	10%	22%	29%	15%	10%	14%	8%	22%	27%	12%	7%	9%
Process													
Agile	14	13%	12%	41%	13%	8%	14%	11%	12%	45%	12%	4%	13%
Traditional	118	10%	19%	29%	16%	11%	15%	8%	18%	27%	14%	9%	10%
Initial Size													
<5K	9	12%	25%	30%	10%	8%	16%	9%	22%	28%	7%	3%	9%
5-20K	23	13%	20%	25%	18%	15%	11%	10%	21%	23%	14%	11%	5%
20-50K	35	7%	19%	31%	19%	8%	15%	6%	16%	30%	17%	5%	9%
50-100K	29	13%	15%	30%	14%	9%	18%	7%	15%	29%	12%	7%	16%
100-250K	24	9%	18%	34%	14%	11%	15%	7%	18%	29%	11%	8%	13%
>250K	12	13%	12%	30%	18%	12%	15%	11%	12%	28%	20%	14%	13%

Figure 4.12: Final Percent Effort Allocation Within Dataset that Documents Effort in All 6 Phases

4.5.2 Robustness Checks on Percent Schedule Change

One concern with our calculation for percent schedule change is that the variation can end up canceling, to some extent, the true extent of the change. When measuring the mean change, the projects that had a large schedule increase are tempered by the projects that saw a significant reduction in total schedule. The consequence is that we get a false sense of how well we predict schedule. With inspiration from Ryan et. al. (2013), we evaluated schedule change using Mean Absolute Percentage Error (MAPE). This takes the absolute value of the percent change in schedule for each report, eliminating the effect of high and low values canceling each other out. The new MAPE mean value for schedule growth was higher, 59% vs. the original 52%. The median APE (MdAPE) was also higher, 24% vs. the original 17%. Still, the new values were not excessively different than originally calculated.

Next we tested for early phase differences as before, splitting the database into two cohorts with higher or lower schedule growth using the MAPE/MdAPE values as the split. The results are in Tables 4.15 and 4.16. Reassuringly, the results are still significant for both tests. In fact, there are more statistically significant values in the MAPE tests than in the original Means tests; All the OE categories are now significant, the Traditional Process is, and two additional ADs are significant. A similar story plays out with the MdAPE table, which contains many of the same significant categories as the original Median split table.

Table 4.15: Difference in Phase 1+2 Percent Effort, Above and Below MAPE of 59% Growth

	N	Below Mean of 59%	Above Mean of 59%	Mean Delta	t Ratio	p-value
Overall	308	30%	21%	-9%	-3.930	**0.0001**
Service						
Air Force	76	28%	19%	-9%	-2.414	**0.0091**
Army	96	28%	19%	-9%	-1.318	**0.0953**
Navy	137	32%	23%	-9%	-3.454	**0.0004**
Super Domain						
AIS	25	36%	19%	-17%	-2.369	**0.0135**
ENG	39	30%	16%	-14%	-2.743	**0.0047**
MS	12	39%	29%	-10%	-0.947	0.1829
RT	233	28%	21%	-7%	-2.732	**0.0034**
Application Domain						
C&C	66	25%	22%	-3%	-0.767	0.2228
CAS	11	42%	19%	-23%	-2.505	**0.0168**
COM	22	38%	31%	-7%	-0.719	0.2401
M&F	*1*					
MP	14	30%	20%	-10%	-1.109	0.1454
RTE	54	30%	20%	-10%	-2.143	**0.0184**
S&S	12	31%	9%	-22%	-3.068	**0.0059**
SP	35	27%	31%	4%	0.493	0.6877
SS	20	31%	9%	-22%	-2.161	**0.0222**
TMDE	7	25%	26%	1%	0.116	0.544
TOOL	6	39%	26%	-13%	-0.773	0.2412
TRN	6	39%	31%	-8%	-0.464	0.3335
VC	33	29%	11%	-18%	-2.451	**0.0100**
VP	22	24%	12%	-12%	-1.689	**0.0525**
Operating Env.						
Air	137	26%	21%	-5%	-1.455	**0.074**
Sea	32	30%	18%	-12%	-1.401	**0.0857**
Space	15	19%	10%	-9%	-1.623	**0.0643**
Surface	125	34%	26%	-8%	-2.169	**0.016**
Process						
Agile	206	29%	20%	-9%	-3.73	**0.0001**
Traditional	103	29%	24%	-5%	-1.402	**0.082**
Initial Size						
<5K	52	26%	14%	-12%	-1.623	**0.0565**
5-20K	97	31%	21%	-10%	-2.713	**0.0040**
20-50K	80	29%	20%	-9%	-2.487	**0.0075**
50-100K	40	33%	25%	-8%	-1.116	0.1358
100-250K	28	30%	31%	1%	0.089	0.5352
>250K	12	29%	25%	-4%	-0.295	0.3867

Table 4.16: Difference in Phase 1+2 Percent Effort, Above and Below MdAPE of 24% Growth

	N	Below Median of 24%	Above Median of 24%	Median Delta	t Ratio	p-value
Overall	308	29%	25%	-4%	-2.158	**0.0159**
Service						
Air Force	76	28%	21%	-7%	-2.115	**0.0189**
Army	96	28%	26%	-2%	-0.319	0.3751
Navy	137	32%	26%	-6%	-2.004	**0.0235**
Super Domain						
AIS	24	34%	30%	-4%	-0.697	0.2466
ENG	39	33%	19%	-14%	-2.854	**0.0035**
MS	12	35%	34%	-1%	-0.070	0.4727
RT	233	28%	25%	-3%	-1.320	**0.0935**
Application Domain						
C&C	66	28%	21%	-7%	-1.599	**0.0574**
CAS	11	44%	28%	-16%	-2.089	**0.0332**
COM	22	33%	37%	4%	0.438	0.6669
M&F	*1*					
MP	14	24%	31%	7%	0.890	0.1963
RTE	54	31%	24%	-7%	-1.757	0.8073
S&S	12	33%	20%	-13%	-1.624	**0.0678**
SP	35	24%	33%	9%	1.622	0.9428
SS	20	35%	16%	-19%	-2.673	**0.0078**
TMDE	7	25%	26%	1%	0.116	0.5440
TOOL	6	26%	39%	13%	0.742	0.7504
TRN	6	53%	30%	-23%	-1.150	0.1568
VC	33	29%	21%	-8%	-1.376	**0.0893**
VP	22	20%	21%	1%	0.130	0.5509
Operating Env.						
Air	137	25%	23%	-2%	-0.726	0.2345
Sea	32	25%	29%	4%	0.585	0.7186
Space	15	21%	11%	-10%	-1.767	**0.0478**
Surface	125	34%	29%	-5%	-1.615	**0.0544**
Process						
Agile	206	30%	24%	-6%	-2.286	**0.0118**
Traditional	103	29%	27%	-2%	-0.452	0.3262
Initial Size						
<5K	52	28%	17%	-11%	-1.937	**0.0294**
5-20K	97	31%	24%	-7%	-1.631	**0.0531**
20-50K	80	29%	24%	-5%	-1.457	**0.0746**
50-100K	40	32%	31%	-1%	-0.016	0.4939
100-250K	28	28%	32%	4%	0.675	0.7472
>250K	12	22%	32%	10%	0.831	0.7873

4.5.3 Robustness Checks on Schedule and Effort

To validate the schedule growth results, we ran a few robustness checks on our data using regressions. The first check was to regress schedule growth by each phase to see which ones could be explanatory factors. This regression is shown in Figure 4.13, below. This modeled the percent schedule growth by the percent effort in Phases 1 thru 5. Phase 6 was left out to avoid multi-collinearity. Both the regression model and each of the parameters were significant at $\alpha = 0.05$. The model as a whole had a very small R^2 value, only 0.084, but we were concerned with the significance of the parameters. The effect summary shows that Phase 1 and 2 had the highest logworth in the model, which validates our decision to focus on early phases as explanatory factors in schedule growth.

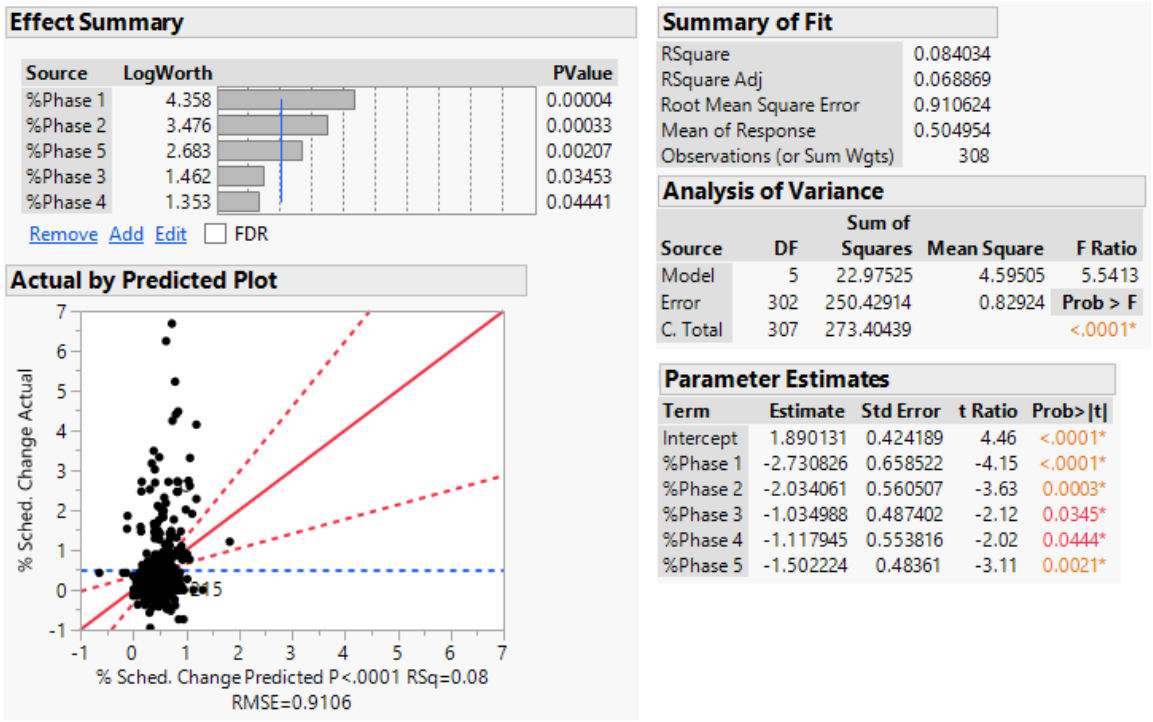


Figure 4.13: Regression of %Phases 1-5 on %Schedule Growth

The second regression model used the combined percent effort in Phase 1+2 and the percent increase in total effort hours. The model and both parameter values were significant at $\alpha = 0.05$. The R^2 value was higher than the first regression, at 0.136, but we still are focused on the individual significance of the parameters. Both the first and second regressions support our findings that schedule and early effort are positively correlated, and that the SRDR data follows this pattern. Additionally, it is critical to note the signs of the "%Phase" parameters, which are all negative. This shows that as more effort is allocated to the first two phases, the percent schedule growth decreases.

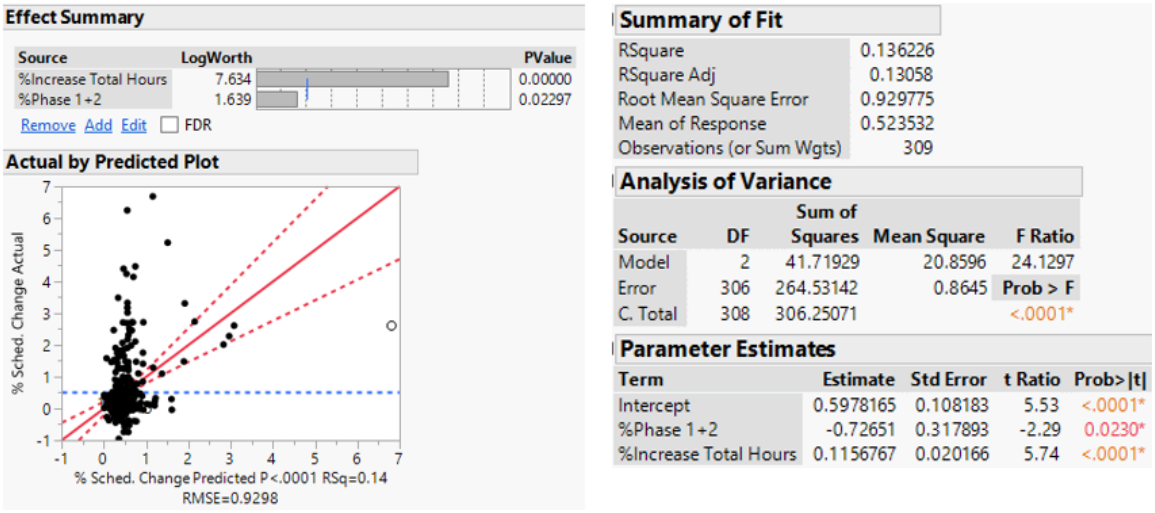


Figure 4.14: Regression of %Phase 1+2 and %Increase Total Hours on %Schedule Growth

V. Conclusion

This section seeks the provide answers pertaining to each of the research questions and validate that the research goals were accomplished. We then discuss limitations of the research, provide recommendations for the DoD software developer, and propose topics for future research.

5.1 Rules of Thumb

Our first research questions asked how the DoD data compared to existing rules of thumb commonly discussed in the field of software development. Unfortunately, due to the high volatility in the data, all mean phase values have a wide confidence interval that nearly encompasses every rule of thumb. It is thus difficult to propose a single rule of thumb, since our results showed that nearly every rule of thumb could be used. Asking this question within a certain category, like Agile projects or Space projects, did not meaningfully reduce said variation in the data.

Realizing that this is not a very helpful answer for a developer or project manager trying to estimate a phasing distribution, we can tenuously recommend to use the rules of thumb by Yang, Borysowich, and Papatheocharous. None of these rules match the mean phase values exactly, but they do hold up relatively well in our LSM tests regardless of categorical filter used. Still, the Yang and Papatheocharous values are awkward for a Rule of Thumb, are not easily remembered, and convey a precision which is inappropriate for our data set. Borysowich's RoT uses dubious ranges for each phase.

To partially alleviate these concerns, we offer a new RoT in Table 5.1 for discourse within DoD programs. Due to the disparity in phase reporting, we had to combine the SRDR phases 5 and 6 for all tests and are unable to recommend a specific breakout

for these phases. The New RoT has the advantage of round numbers for easy recall and application by the practitioner, as well as a smaller LSM magnitude than the Yang rule for most filters on the data. (These LSM magnitude differences are shown in the Appendices.) Though we attempted to correct for phase definitions between researchers, validation issues remain. To ally this concern, we further simplified the New RoT to 30% Analysis & Design, 40% Coding, and 30% Test & Integration.

Table 5.1: Means of Dataset and Proposed Rules of Thumb

	Phase 1	**Phase 2**	**Phase 3**	**Phase 4**	**Phase 5+6**
SRDR Mean	11%	18%	37%	16%	18%
New Rule of Thumb	**15%**	**15%**	**40%**	**20%**	**10%**
Yang	16%	15%	40%	22%	7%
Papatheocharous	9%	14%	42%	18%	7%
Borysowich	15%	15%	30%	15%	25%

Providing this new RoT based solely on the means of our dataset is an incomplete conclusion at best. Within our data are a host of projects that were both well managed and poorly managed, evidenced by our schedule research. A better rule of thumb would focus on the best-performing projects instead of lumping all projects together. To incorporate our schedule results, we found the mean phase percentages for all projects that experienced no schedule growth or negative schedule growth (N=97). The results, shown in Table 5.2, allocate slightly more effort to the first two phases and less to phases 3 and 5+6 than the original mean values.

Table 5.2: Schedule-Adjusted Mean Phase Values

	Phase 1	**Phase 2**	**Phase 3**	**Phase 4**	**Phase 5+6**
SRDR Mean	11%	18%	37%	16%	18%
Schedule-Adjusted RoT	**13%**	**19%**	**35%**	**16%**	**16%**

While the results in the table compliment our schedule results (discussed in the next section), the adjusted phasing numbers again fall victim to their inability to be easily remembered and applied. To improve the usefulness of our RoT we can

simplify further to an even split between Phase 1+2, Phase 3, and Phase 4+5+6, with one-third effort each. This final Rule of Thumb is displayed in Table 5.3.

Table 5.3: Schedule-Adjusted Rule of Thumb

1/3	**Requirements & Design**
1/3	**Coding**
1/3	**Testing & Integration**

An integral observation for this research is that the mean phase values found in our dataset are relatively similar to the mean phase values in the Yang and Papatheocharous studies. This is a unique result, since there are no overlapping projects between the three databases studied. In view of this similarity, and the fact that our dataset was the largest of the three empirical studies, it is likely that our schedule and effort change results are statistically applicable to the software industry at large and not limited to the DoD flavor of software development.

5.2 Phasing Change from Estimate to Result

Our analysis showed minimal change in the effort distribution across each phase between the initial estimate and the final result. At the same time, the average size of the software and the average total effort allocated to the software increased. This means that even though total effort increases the phasing distribution does not necessarily change; the effort increases proportionally in each phase.

Table 5.4: Overall Percent Effort Change from Initial to Final, by Phase

	1	**2**	**3**	**4**	**5**	**6**
Overall	0%	1%	1%	0%	-2%	-1%

For the overall SRDR dataset, the average change at each phase did not exceed 2 percent. (See Table 5.4). This result indicates that there is strong dependency

between the phases even when changes or modifications are happening to the program. This makes it all the more important to get the estimated phasing correct in the estimate, the consequences of which are elaborated in the next section.

5.3 Schedule

The results show that final projects with a larger than average schedule growth allocated less effort in the first two phases of development, when compared to projects with less than average growth. On the opposite side, the projects that did allocate a larger proportion of effort in the first two phases experienced less schedule growth or negative schedule growth. Of the 26 categories tested, 16 categories had significantly different phasing distributions between the two schedule growth cohorts. The cohorts broken out by Service were significant, as were 3/4 of the Super Domains. The cohorts were significant for projects with an ESLOC $< 50K$. Additionally, the growth cohorts within Agile software were highly significant, so Agile projects in particular should focus on early phase effort. Application domain did not trend significantly, with less than half the application domains testing significant. This is in line with the literature (Amato, 2021).

This relationship between schedule growth and early phasing effort is backed by the literature, which has shown that the errors that most contribute to growth are commonly the result of rushed or inadequate effort in early phases (Boehm, 1987; Thibodeau and Dodson, 1980). Based on our results, we find it prudent to increase early-phase effort allocation and focus on fully defining the program before beginning the coding process.

5.4 Recommendations

The goal of this thesis is to provide beneficial analysis for the practitioner in DoD Software Development. Our general recommendations based on the empirical results herein are the following: first, that the Air Force Cost Analysis Agency augments its next edition of the Software Development Cost Estimating Guidebook with our schedule-adjusted rule of thumb, allocating equal effort between Requirements & Design, Coding, and Testing & Integration. Second, we recommend that projects allocate more effort in the first two phases (around 30 percent), especially for small projects. This is in line with our proposed rule of thumb and is correlated with less schedule growth. Third, we recommend that programs experiencing schedule slip in early phases should not attempt to stay on schedule for subsequent phases. Doing so will likely cause further schedule slip due to incomplete analysis and design work affecting the work in coding and testing, invoking the Pareto rule.

5.4.1 Limitations

The recommendations put forth are not intended to be followed without caution and care. The research faced multiple limitations, enumerated and elaborated here:

1. The variation in the data left much to be desired. We were unable to provide a strong recommendation for a rule of thumb, given that nearly every RoT fit the data to some degree. We predict there are unique methods to control for variation in the dataset that would produce a significant Hotelling Test result, possibly in future research.

2. The definitions of the phases were not a perfect match when comparing between the SRDR phases and the Rules of Thumb. The RoT have been developed at various points in the last 70 years, based on disparate software development

styles, nomenclature, programming languages, and more. While every effort was made to compare apples-to-apples, there can never be a perfect match.

3. Overlap of work between phases and differences in how contractors input data contribute to possibly misconstrued numbers in phase effort allocations. A large portion of the reports studied were missing effort for at least one phase.

4. Only a small subset of the SRDR database was evaluated in this research. Out of over five thousand reports, we were only able to consider roughly 300 pairs for our schedule analysis. The rejected reports could have a different phasing profile or a different amount of schedule growth than the sample we used.

5.4.2 Future Research

The end of this thesis should mark the beginning of a new focus on phase relationship research within the DoD. Historically, there has not been a focus on how each phase impacts the next or impacts characteristics of the end result. We recommend the following questions to be considered for future research.

First, does the data follow a Rayleigh curve? And what are the parameters of the associated Rayleigh equation? Is there a measurable relationship between the initial estimate's Rayleigh parameters and the final cost and schedule? We were unable to incorporate Rayleigh analysis with our results due to a lack of time, but the concept remains an intriguing one. A further analysis would also look at non-Rayleigh models, such as Weibull curves.

Second, how do the phases relate to cost? We considered effort and schedule change as a proxy for cost, but future theses could directly analyze the program costs and how they are affected by changes in effort. This would provide a better real-world estimate for cost estimation models.

VI. Appendix A - Mean Final Effort by Phase

Table 6.1: Mean Final Effort by Phase (rounded to nearest 100 SLOC)

	N	Phase 1	Phase 2	Phase 3	Phase 4	Phase 5	Phase 6
Full Dataset	722	5800	9600	14600	9200	6800	6700
Service							
Air Force	230	4400	10000	13200	9500	7300	8800
Army	216	6900	13400	16900	9600	6600	3600
Navy	276	5900	7300	14300	8700	6500	6500
Super Domain							
AIS	48	2700	9500	18300	5600	1600	1900
ENG	114	4900	11100	18200	8900	3200	5100
MS	26	1300	2900	4900	900	1000	1100
RT	534	6400	9700	14000	9900	8100	7900
OE - Summary							
Air Vehicle	268	5100	8700	12900	9000	7400	9000
Sea System	78	5400	5600	10100	5800	6400	1300
Space System	48	5300	24000	22200	19600	8200	9200
Surface	339	6500	9700	16100	9300	6200	4700
Process							
Agile	123	8400	5200	10100	5600	5400	7400
Traditional	595	5300	10400	15300	9700	7000	6600
Initial Size							
<5K	104	500	700	1200	1000	900	800
5-20K	188	1400	2600	4000	2400	2800	1300
20-50K	174	2300	6800	9500	7400	4900	3800
50-100K	113	4700	9100	18100	10200	7400	9500
100-250K	105	16700	20200	34700	20900	16100	13500
>250K	38	26800	46900	64600	42100	20500	20700

VII. Appendix B - Summary Statistics for Cited Empirical Phase Research

Phase	Plan&Req.	Design	Code	Test	Trans.
Min	1.82%	0.62%	6.99%	4.24%	0.06%
Max	35%	50.35%	92.84%	50.54%	36.45%
Median	15.94%	14.21%	36.36%	19.88%	4.51%
Mean	16.14%	14.88%	40.36%	21.57%	7.06%
Stdev	8.62%	8.91%	16.82%	11.04%	7.06%

Figure 7.1: Summary Statistics for Yang et. al. Research (2008)

	Summary Work Effort	Plan	Specify	Design	Build	Test	Imple ment	Unpha sed
N	65	65	65	65	65	65	65	65
Max	47252	10800	6500	5807	30000	6966	5393	14382
Min	172	4	4	17	70	40	2	0
Med	4294	158	250	400	1300	626	168	148
Mean	6999.71	575.63	556.23	831.77	2577.68	1085.55	390.38	982.46
Std. Dev.	8451.03	1456.60	1026.80	1024.79	4686.74	1364.81	784.02	2279.87
25%	2337	64.50	98	198	570.50	267	49	0
75%	7843	488.50	669.50	1080	2661.50	1411	392.50	563

Figure 7.2: Summary Statistics for Papatheocharous Research (2017)

VIII. Appendix C - Fractional Standard Deviation Method

This is a descriptive section that attempts to visually pinpoint which RoT can meet various accuracy targets based on fractions of standard deviation from the mean of the phase data. Table 8.1 shows the mean and standard deviation for each phase, by Service. Each rule's prescribed value for each phase is listed in the table columns, broken out by service. The cell's highlighted colors are conditional on if the RoT value fits within a target range based on the standard deviation. Dark green and bold text apply when the RoT value is within the mean phase effort ± one-fourth of the standard deviation. This is the smallest and hardest target to hit and reveals which rules have a central tendency to the mean. The second, larger target is denoted by Pale Green highlight, and applies when the RoT value falls within ± one-half the standard deviation from the mean. All RoT values that fall outside of one-half a standard deviation have missed the mark and are left blank.

Service	Phase	Mean	StDev	Yang	Borysowich	Papatheocharous	Heijstek	Sommerville	Zelkowitz	Ambler	Phase	Mean	StDev	Thibodeau	Brooks	Bennington	Boehm
Air Force	1	0.077	0.086	0.16	0.15	0.19	0.22	0.15	0.10	0.10							
	2	0.187	0.180	**0.15**	**0.15**	0.14	0.11	0.25	0.25	0.25	1+2	0.264	0.195	0.40	0.33	0.50	0.60
	3	0.402	0.242	**0.40**	0.30	**0.42**	**0.38**	0.20	0.20	0.55	3	0.402	0.242	0.20	0.17	0.10	0.15
	4+5+6	0.334	0.219	**0.29**	0.40	0.25	**0.29**	0.40	0.45	0.10	4+5+6	0.334	0.219	0.40	0.50	0.40	0.25
Army	1	0.159	0.207	**0.16**	**0.15**	**0.19**	0.22	**0.15**	0.10	0.10							
	2	0.183	0.211	**0.15**	**0.15**	**0.14**	0.11	0.25	0.25	0.25	1+2	0.342	0.263	**0.40**	**0.33**	0.50	0.60
	3	0.328	0.178	0.40	**0.30**	0.42	0.38	0.20	0.20	0.55	3	0.328	0.178	0.20	0.17	0.10	0.15
	4+5+6	0.330	0.213	**0.29**	0.40	0.25	**0.29**	0.40	0.45	0.10	4+5+6	0.330	0.213	0.40	0.50	0.40	0.25
Navy	1	0.109	0.122	0.16	0.15	0.19	0.22	0.15	**0.10**	**0.10**							
	2	0.176	0.118	**0.15**	**0.15**	0.14	0.11	0.25	0.25	0.25	1+2	0.285	0.160	0.40	0.33	0.50	0.60
	3	0.363	0.208	**0.40**	0.30	0.42	**0.38**	0.20	0.20	0.55	3	0.363	0.208	0.20	0.17	0.10	0.15
	4+5+6	0.352	0.208	0.29	**0.40**	0.25	0.29	**0.40**	0.45	0.10	4+5+6	0.352	0.208	**0.40**	0.50	**0.40**	0.25

Highlighted Target Bounds		
+/- StDev/4	+/- StDev/2	Outside of StDev/2

Table 8.1: Standard Deviation Model by Service

As stated earlier, phases 4 thru 6 were combined to allow comparison across all RoT and minimize errors found in the dataset. The left side of the table contains seven

rules of thumb that separate out the effort allocation for phases 1 and 2. Referencing Figure 3.1, this is test sequence **"B"**. The four RoT on the right side consider all work before the main coding phase as one phase of effort and consequently only prescribe one effort percentage for the first two phases. These RoT use test sequence **"C"**. Due to this, the right side of the table combines the first two phases and recalculates the standard deviation. The mean and stdev for the third phase and the combined 4+5+6 phase remain the same as the left side of the table.

The RoT columns are roughly ordered by overall fit, with the left-most RoT matching the data the best, and the right-most RoT consistently in the outfield. RoT columns with more dark green highlights (and bold text) indicate a better fit for the data, having hit more of the smallest targets. The Yang RoT had the best fit for all services, landing within 1/4th of a standard deviation from the mean in 8 out of 12 rows. Borysowich was second-best, with 6 of 12 rows in the smallest target range. The alternative phase schemes of Thibodeau, Brooks, Bennington, and Boehm were not as accurate in estimating the mean as the best four phases on the left side of the table. Boehm fared the worst and did not come within 1/4th a standard deviation in any phase, for any service.

We repeated this analysis based on Operating Environment (OE) and Super Domain (SD) and provide the results in Table 8.2. Similar results occur. Yang and Borysowich perform about the same, though Yang does hit the small 1/4 StDev target less than in Table 8.1. Zelkowitz and Amber perform marginally better than when evaluated against the Services, hitting the smallest target for at least one phase in each category of OE and SD.

As before, the right four RoT with the combined phase 1+2 do not stand out as better options than Yang or Borysowich. The exception is for the SD of MS (Mission Support), where Thibodeau performs on par with the first five RoT on

		Phase	Mean	StDev	Yang	Borysowich	Papatheocharous	Heijstek	Sommerville	Zelkowitz	Ambler	Phase	Mean	StDev	Thibodeau	Brooks	Bennington	Boehm
Operating Environment	Air Vehicle	1	0.088	0.090	0.16	0.15	0.19	0.22	0.15	**0.10**	**0.10**							
		2	0.158	0.169	**0.15**	**0.15**	**0.14**	0.11	0.25	0.25	0.25	1+2	0.246	0.187	0.40	0.33	0.50	0.60
		3	0.388	0.227	**0.40**	0.30	**0.42**	**0.38**	0.20	0.20	0.55	3	0.388	0.227	0.20	0.17	0.10	0.15
		4+5+6	0.366	0.211	0.29	**0.40**	0.25	0.29	**0.40**	0.45	0.10	4+5+6	0.366	0.211	**0.40**	0.50	**0.40**	0.25
	Sea System	1	0.088	0.159	0.16	0.15	0.19	0.22	0.15	**0.10**	**0.10**							
		2	0.183	0.126	0.15	0.15	0.14	0.11	0.25	0.25	0.25	1+2	0.271	0.203	0.40	0.33	0.50	0.60
		3	0.374	0.192	**0.40**	0.30	**0.42**	**0.38**	0.20	0.20	0.55	3	0.374	0.192	0.20	0.17	0.10	0.15
		4+5+6	0.355	0.231	0.29	**0.40**	0.25	0.29	**0.40**	0.45	0.10	4+5+6	0.355	0.231	**0.40**	0.50	**0.40**	0.25
	Space System	1	0.104	0.087	0.16	0.15	0.19	0.22	0.15	**0.10**	**0.10**							
		2	0.141	0.148	**0.15**	**0.15**	**0.14**	**0.11**	0.25	0.25	0.25	1+2	0.245	0.169	0.40	0.33	0.50	0.60
		3	0.388	0.224	**0.40**	0.30	**0.42**	**0.38**	0.20	0.20	0.55	3	0.388	0.224	0.20	0.17	0.10	0.15
		4+5+6	0.368	0.236	0.29	**0.40**	0.25	0.29	**0.40**	0.45	0.10	4+5+6	0.368	0.236	**0.40**	0.50	**0.40**	0.25
	Surface	1	0.141	0.179	**0.16**	**0.15**	0.19	0.22	**0.15**	**0.10**	**0.10**							
		2	0.206	0.176	0.15	0.15	0.14	0.11	0.25	0.25	0.25	1+2	0.347	0.215	**0.40**	**0.33**	0.50	0.60
		3	0.342	0.204	0.40	**0.30**	0.42	**0.38**	0.20	0.20	0.55	3	0.342	0.204	0.20	0.17	0.10	0.15
		4+5+6	0.311	0.205	**0.29**	0.40	0.25	**0.29**	0.40	0.45	0.10	4+5+6	0.311	0.205	0.40	0.50	0.40	0.25
Super Domain	AIS	1	0.090	0.068	0.16	0.15	0.19	0.22	0.15	**0.10**	**0.10**							
		2	0.212	0.176	0.15	0.15	0.14	0.11	**0.25**	**0.25**	**0.25**	1+2	0.301	0.188	0.40	**0.33**	0.50	0.60
		3	0.441	0.213	**0.40**	0.30	**0.42**	0.38	0.20	0.20	0.55	3	0.441	0.213	0.20	0.17	0.10	0.15
		4+5+6	0.258	0.212	**0.29**	0.40	**0.25**	**0.29**	0.40	0.45	0.10	4+5+6	0.258	0.212	0.40	0.50	0.40	**0.25**
	ENG	1	0.086	0.123	0.16	0.15	0.19	0.22	0.15	**0.10**	**0.10**							
		2	0.206	0.221	0.15	0.15	0.14	0.11	**0.25**	**0.25**	**0.25**	1+2	0.292	0.237	0.40	**0.33**	0.50	0.60
		3	0.353	0.196	**0.40**	0.30	0.42	**0.38**	0.20	0.20	0.55	3	0.353	0.196	0.20	0.17	0.10	0.15
		4+5+6	0.356	0.239	0.29	**0.40**	0.25	0.29	**0.40**	0.45	0.10	4+5+6	0.356	0.239	**0.40**	0.50	**0.40**	0.25
	MS	1	0.180	0.215	**0.16**	**0.15**	**0.19**	**0.22**	**0.15**	0.10	0.10							
		2	0.219	0.232	0.15	0.15	0.14	0.11	**0.25**	**0.25**	**0.25**	1+2	0.399	0.268	**0.40**	0.33	0.50	0.60
		3	0.300	0.235	0.40	**0.30**	0.42	0.38	0.20	0.20	0.55	3	0.300	0.235	0.20	0.17	0.10	0.15
		4+5+6	0.301	0.248	**0.29**	0.40	**0.25**	**0.29**	0.40	0.45	0.10	4+5+6	0.301	0.248	0.40	0.50	0.40	**0.25**
	RT	1	0.117	0.150	0.16	**0.15**	0.19	0.22	**0.15**	**0.10**	**0.10**							
		2	0.172	0.151	**0.15**	**0.15**	**0.14**	0.11	0.25	0.25	0.25	1+2	0.289	0.198	0.40	**0.33**	0.50	0.60
		3	0.365	0.215	**0.40**	0.30	0.42	**0.38**	0.20	0.20	0.55	3	0.365	0.215	0.20	0.17	0.10	0.15
		4+5+6	0.345	0.205	0.29	0.40	0.25	0.29	0.40	0.45	0.10	4+5+6	0.345	0.205	0.40	0.50	0.40	0.25

Highlighted Target Bounds		
+/- StDev/4	+/- StDev/2	Outside of StDev/2

Table 8.2: Standard Deviation Model by Operating Env. (top) and Super Domain (bottom)

the left half. Incidentally, Thibodeau and Sommerville have the same RoT, though Sommerville splits his 40 percent allocation for phases 1 and 2 into 15 and 25 percent respectively. This means that under many scenarios, Thibodeau and Sommerville will return similar results when trying to match the RoT to data.

The next table, 8.3, shows the standard deviation target results based on Development Process. No RoT is the best fit for both Agile and Traditional processes. For Agile development, Papatheocharous and Heijstek match the data best. For Traditional development, Borysowich is the best fit, with Yang in second place. This finding partially meshes with the LSM Models by Process, where Papatheocharous is

the best fit for Agile, but Yang is the best fit for Traditional. The four RoT on the right side of the table were generally poor fits.

Process		Phase	Mean	StDev	Yang	Borysowich	PaPatheocharous	Heijstek	Sommerville	Zelkowitz	Ambler	Phase	Mean	StDev	Thibodeau	Brooks	Bennington	Boehm
	Agile	1	0.200	0.236	**0.16**	**0.15**	**0.19**	**0.22**	**0.15**	0.10	0.10							
		2	0.136	0.132	**0.15**	**0.15**	**0.14**	**0.11**	0.25	0.25	0.25	1+2	0.349	0.258	**0.40**	**0.33**	0.50	0.60
		3	0.432	0.291	**0.40**	0.30	**0.42**	**0.38**	0.20	0.20	0.55	3	0.413	0.295	0.20	0.17	0.10	0.15
		4+5+6	0.233	0.223	0.29	0.40	**0.25**	**0.19**	0.40	**0.25**	0.10	4+5+6	0.238	0.214	0.40	0.50	0.40	**0.25**
	Traditional	1	0.098	0.111	0.16	0.15	0.19	0.22	0.15	**0.10**	**0.10**							
		2	0.187	0.160	**0.15**	**0.15**	0.14	0.11	0.25	0.25	0.25	1+2	0.285	0.194	0.40	**0.33**	0.50	0.60
		3	0.360	0.197	**0.40**	0.30	0.42	**0.38**	0.20	0.20	0.55	3	0.356	0.194	0.20	0.17	0.10	0.15
		4+5+6	0.355	0.206	0.29	**0.40**	0.25	0.19	**0.40**	0.25	0.10	4+5+6	0.359	0.208	**0.40**	0.50	**0.40**	0.25

Highlighted Target Bounds

+/- StDev/4	+/- StDev/2	Outside of StDev/2

Table 8.3: Standard Deviation Model by Process

The last table shows the standard deviation target results by Size, measured in ESLOC. Overall, Yang appears to match the data the best in most categories. Borysowich matches the data better for projects less than 5K ESLOC. The four RoT on the right side of the table were poor fits overall, though they occasionally performed on par or better than Sommerville, Zelkowitz, and Amber.

At the end of this inquiry, a couple observations are worth noting. One, the phase 3 mean was the most difficult to hit for most RoT. Depending on categorical filter, either 6 or 7 rules failed to come within a half standard deviation of the mean, leaving the cell blank. No other phase was as difficult to fit to, especially for the right seven RoT after Heijstek. Out of 19 categorical rows, only three separate instances occurred where a RoT came within a half stddev of the phase 3 mean.

Two, though the left four RoT were overall the best, they missed the Phase 1 targets entirely in three instances: Service = Air Force, OE = Air Vehicle or Space System, and SD = AIS. In these instances, the Phase 1 mean value was roughly half the RoT phase 1 values, with a relatively small variance. Additionally, there's a logical correlation between these categories, since the Air Force comprises a large

Size (ESLOC)	Phase	Mean	StDev	Yang	Boraysowich	Papatheocharous	Heijstek	Sommerville	Zelkowitz	Ambler	Phase	Mean	StDev	Thibodeau	Brocks	Bennington	Boehm
<5K	1	0.129	0.167	**0.16**	**0.15**	0.19	0.22	**0.15**	**0.10**	**0.10**							
	2	0.136	0.143	**0.15**	**0.15**	**0.14**	**0.11**	0.25	0.25	0.25	1+2	0.265	0.225	0.40	0.33	0.50	0.60
	3	0.341	0.235	**0.40**	**0.30**	0.42	**0.38**	0.20	0.20	0.55	3	0.341	0.235	0.20	0.17	0.10	0.15
	4+5+6	0.394	0.249	0.29	**0.40**	0.25	0.19	**0.40**	0.25	0.10	4+5+6	0.394	0.249	**0.40**	0.50	**0.40**	0.25
5-20K	1	0.111	0.149	0.16	0.15	0.19	0.22	0.15	**0.10**	**0.10**							
	2	0.188	0.178	**0.15**	**0.15**	0.14	0.11	0.25	0.25	0.25	1+2	0.299	0.209	0.40	**0.33**	0.50	0.60
	3	0.373	0.228	**0.40**	0.30	**0.42**	**0.38**	0.20	0.20	0.55	3	0.373	0.228	0.20	0.17	0.10	0.15
	4+5+6	0.328	0.215	**0.29**	0.40	0.25	0.19	0.40	0.25	0.10	4+5+6	0.328	0.215	0.40	0.50	0.40	0.25
20-50K	1	0.091	0.121	0.16	0.15	0.19	0.22	0.15	**0.10**	**0.10**							
	2	0.199	0.169	0.15	0.15	0.14	0.11	0.25	0.25	0.25	1+2	0.290	0.206	0.40	**0.33**	0.50	0.60
	3	0.364	0.183	**0.40**	0.30	0.42	**0.38**	0.20	0.20	0.55	3	0.364	0.183	0.20	0.17	0.10	0.15
	4+5+6	0.346	0.204	0.29	0.40	0.25	0.19	0.40	0.25	0.10	4+5+6	0.346	0.204	0.40	0.50	0.40	0.25
50-100K	1	0.102	0.116	0.16	0.15	0.19	0.22	0.15	**0.10**	**0.10**							
	2	0.167	0.114	**0.15**	**0.15**	**0.14**	0.11	0.25	0.25	0.25	1+2	0.270	0.164	0.40	0.33	0.50	0.60
	3	0.395	0.191	**0.40**	0.30	**0.42**	**0.38**	0.20	0.20	0.55	3	0.395	0.191	0.20	0.17	0.10	0.15
	4+5+6	0.336	0.200	**0.29**	0.40	0.25	0.19	0.40	0.25	0.10	4+5+6	0.336	0.200	0.40	0.50	0.40	0.25
100-250K	1	0.136	0.159	**0.16**	**0.15**	0.19	0.22	**0.15**	**0.10**	**0.10**							
	2	0.189	0.198	**0.15**	**0.15**	**0.14**	0.11	0.25	0.25	0.25	1+2	0.325	0.217	0.40	**0.33**	0.50	0.60
	3	0.349	0.221	**0.40**	**0.30**	0.42	**0.38**	0.20	0.20	0.55	3	0.349	0.221	0.20	0.17	0.10	0.15
	4+5+6	0.326	0.207	**0.29**	0.40	0.25	0.19	0.40	0.25	0.10	4+5+6	0.326	0.207	0.40	0.50	0.40	0.25
>250K	1	0.147	0.196	**0.16**	**0.15**	**0.19**	0.22	**0.15**	**0.10**	**0.10**							
	2	0.198	0.208	**0.15**	**0.15**	0.14	0.11	**0.25**	**0.25**	**0.25**	1+2	0.346	0.218	**0.40**	**0.33**	0.50	0.60
	3	0.356	0.256	**0.40**	**0.30**	**0.42**	**0.38**	0.20	0.20	0.55	3	0.356	0.256	0.20	0.17	0.10	0.15
	4+5+6	0.298	0.194	**0.29**	0.40	**0.25**	0.19	0.40	**0.25**	0.10	4+5+6	0.298	0.194	0.40	0.50	0.40	**0.25**

Highlighted Target Bounds		
+/- StDev/4	+/- StDev/2	Outside of StDev/2

Table 8.4: Standard Deviation Model by Size

portion of the software projects related to Air Vehicles and Space Systems.

In summary, the standard deviation targets, as shown, is not a statistical test. While we initially considered a traditional ANVOA test, the variation in the data meant results would be non-significant. The standard deviation model assists in answering which RoT might be more useful than others, though these results should be taken with a substantial helping of salt.

IX. Appendix D - New Rule of Thumb Results

Table 9.1: Comparison of LSM Results for New Rule of Thumb to Yang

	New	**Yang**
Full Dataset	**0.0707**	0.0825
Army	0.0975	**0.086**
Air Force	**0.0812**	0.098
Navy	**0.0671**	0.097
Super Domain		
RT	**0.0707**	0.0825
MS	0.1049	**0.102**
ENG	**0.1122**	0.1241
AIS	**0.097**	0.097
Operating Env.		
Surface	**0.0812**	0.086
Space	**0.1208**	0.1342
Sea	**0.0762**	0.0894
Air	**0.0787**	0.09273
Process		
Traditional	**0.0906**	0.102
Agile	0.0812	**0.0678**
Size		
<5K	**0.0970**	0.1105
5-20K	**0.0927**	0.1049
20-50K	**0.1095**	0.1225
50-100K	**0.0245**	0.0316
100-250K	**0.0894**	0.0990
>250K	0.1393	**0.1342**

X. Appendix E - Correlation Between Schedule and Effort

Correlations

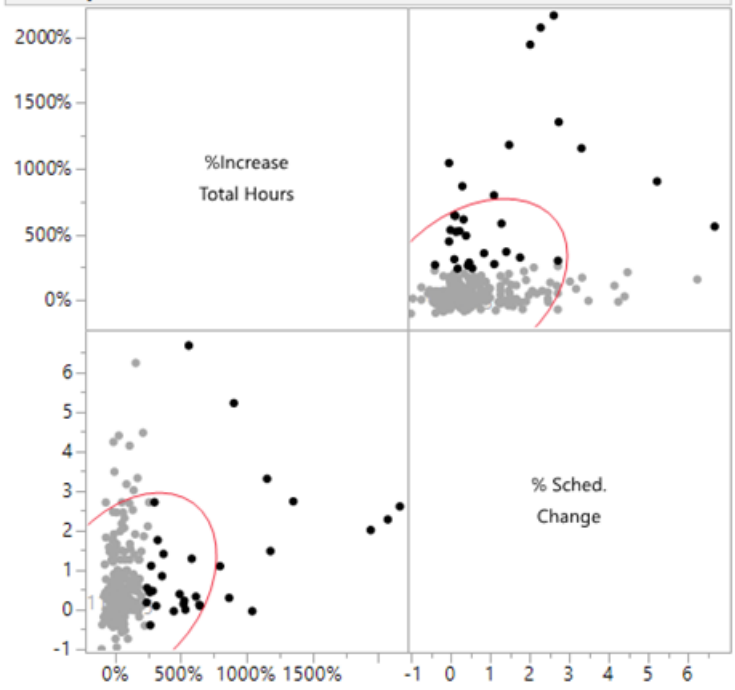
	%Increase Total Hours	% Sched. Change
%Increase Total Hours	1.0000	0.3484
% Sched. Change	0.3484	1.0000

There are 1 missing values. The correlations are estimated by REML method.

Correlation Probability

	%Increase Total Hours	% Sched. Change
%Increase Total Hours	<.0001	<.0001
% Sched. Change	<.0001	<.0001

Scatterplot Matrix



Multivariate Simple Statistics

Column	N	DF	Mean	Std Dev	Sum	Minimum	Maximum
%Increase Total Hours	309	308.00	1.0633	2.7213	329.337	-0.7877	21.6396
% Sched. Change	310	309.00	0.5208	0.9967	161.461	-0.7364	6.6848

Bibliography

AFCAA; NCCA. (2008). *Software Development Cost Estimating Guidebook* (Vol. 1). Software Technology Support Center.

Amato, E. P. (2021). An Analysis of Application Type , Super Domain , and Productivity in Software Intensive Defense Acquisition.

Ambler, S. W. (2005). *A manager's introduction to the Rational Unified Process (RUP)* (tech. rep. March).

Benington, H. D. (1983). Production of Large Computer Systems. *Annals of the History of Computing*, *5*(4), 350–361.

Blalock, C. D. (1988). *An Analysis Of Schedule Determination In Software Program Development And Software Development Estimation Models* (Doctoral dissertation). AFIT.

Boehm, B. (1987). Industrial Software Metrics Top 10 List. *IEEE Software*, *4*(5), 84. https://doi.org/10.1109/MS.1987.231780

Boehm, B. (2000). *Acknowledgments* (tech. rep.). University of Southern California (USC)
us.

Boehm, B., & Basili, V. (2001). Top 10 list [software development]. *IEEE Computer*, *34*(1), 135–137. https://doi.org/10.1109/2.962984

Borysowich, C. (2005). Effort Distribution Across the Software Lifecycle. *TOOLBOX.tech*. https://www.toolbox.com/tech/enterprise-software/blogs/effort-distribution-across-the-software-lifecycle-102605/

Brooks, F. P. (1975). The Mythical Man-Month, 193. https://doi.org/10.1145/800027.808439

Brown, G. (2015). *Accuracy of Time Phasing Aircraft Development Using the Continuous Distribution Function* (Doctoral dissertation). AFIT.

Brown, T., Gallagher, M., & White, E. (2002). Weibull-based Forecasting of R&D Program Budgets. *Cost Analysis & Management*, *4*(1), 41–54.

Coggins, G. A. (1993). Software cost Estimating Models: A Comparative Study of What the Models Estimate
Explores schedule compressions, pg 80. This is a motive for good planning of appropriate efforts. Two schools: miin dev effort, min dev time (higher costs). For schedule changes, models either increase or decrease expected effort (what?????) - All models say it increases in both cases, but SEER-SEM assumes total effort decreases when sched. increases - based on optimal staffing rates (longer time, staffing can be lowered, = lower $$).

Daly, B. (1990). *Comparison Of Software Schedule Estimators* (Doctoral dissertation). AFIT.

D'Amico, C. N. (2017). A Longitudinal Study and Color Rating System of Acquisition Cost Growth.

Dayal Chauhan, B., Rana, A., & Sharma, N. K. (2018). Impact of development methodology on cost & risk for development projects. *2017 6th International*

Conference on Reliability, Infocom Technologies and Optimization: Trends and Future Directions, ICRITO 2017, *2018-Janua*, 267–272. https://doi.org/10.1109/ICRITO.2017.8342436

Everett, G. D., & McLeod Jr., R. (2007). *Software Testing Software Development Life Cycle* (Vol. 5).

GAO. (2021). Software Development: DOD Faces Risks and Challenges in Implementing Modern Approaches and Addressing Cybersecurity Practices. (June).

Goljan, J. (2021). Developing a Cost of Delay (CoD) Framework for the DoD & Analyzing the Current State of Air Force Agile Cost Estimation.

Heijstek, W., & Chaudron, M. R. V. (2008). Effort distribution in model-based development. *Proceedings of the 2nd Workshop on Model Size Metrics, Nashville, TN*.

Jolak, R., Ho-Quang, T., Chaudron, M., & Shiffelers, R. (2018). Teaching model-based software engineering. *Models '18*. https://doi.org/AChttps://doi.org/10.1145/3239372.3239404

Lanham, N., Russo, M., Strickland, D., Cipressi, R., Palmer, S., & Rudloff, C. (2018). *Software Resource Data Report (SRDR) Verification And Validation (V & V) Guide* (tech. rep.).

Papatheocharous, E., Bibi, S., Stamelos, I., & Andreou, A. S. (2017). An investigation of effort distribution among development phases: A four-stage progressive software cost estimation model. *Journal of Software: Evolution and Process*, *29*(10). https://doi.org/10.1002/smr.1881

Prasetya, K. D., Suharjito, & Pratama, D. (2021). Effectiveness Analysis of Distributed Scrum Model Compared to Waterfall approach in Third-Party Application Development. *Procedia Computer Science*, *179*(2019), 103–111. https://doi.org/10.1016/j.procs.2020.12.014

Pressman, R. (2001). *Software Engineering: a Practitioner's Approach* (5th). McGraw-Hill.

Putnam, L. H. (1978). A General Empirical Solution to the Macro Software Sizing and Estimating Problem. *IEEE Transactions on Software Engineering*, *SE-4*(4), 345–361. https://doi.org/10.1109/TSE.1978.231521

Putnam, L. H. (1991). Putnam_Chapter2.pdf.

Putnam, L. H., & Myers, W. (1992). *Measures for Excellence*. Prentice Hall.

Roza, D. (2021). Air Force Software is so bad the guy in charge it all is about to quit. https://taskandpurpose.com/news/air-force-cybersecurity-nicolas-chaillan/

Schumacker, R. (2016). *Using R with Multivariate Statistics*. SAGE Publications. https://us.sagepub.com/sites/default/files/upm-assets/70364%7B%5C_%7Dbook%7B%5C_%7Ditem%7B%5C_%7D70364.pdf

Sommerville, I. (2016). *Software engineering (10th edition)*.

Taipale, e. a. (2006). Factors affecting software testing time schedule. *Proceedings of the Australian Software Engineering Conference, ASWEC*, *2006*, 283–291. https://doi.org/10.1109/ASWEC.2006.27

Tan, T. (2012). Domain-Based Effort Distribution Model for Software Cost Estimation. (August).

Tan, T., Boehm, B., & Clark, B. (2011). *An Investigation on Application Domains for Software Effort Distribution Patterns* (tech. rep.). Center for Systems and Software Engineering, USC.

The Standish Group, I. (2015). *CHAOS Report 2015* (tech. rep.).

Thibodeau, R., & Dodson, E. N. (1980). Life cycle phase interrelationships. *The Journal of Systems and Software*, *1*(100), 203–211. https://doi.org/10.1016/0164-1212(79)90021-9

Unger, E. (2001). *Relating Initial Budget To Program Growth With Rayleigh And Weibull Models* (Doctoral dissertation). AFIT.

Violette, T. (2021). *An Analysis of Stability in SRDR CSCIs* (Doctoral dissertation).

Yang, Y., He, M., Li, M., Wang, Q., & Boehm, B. (2008). Phase distribution of software development effort. *ESEM'08: Proceedings of the 2008 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 61–69. https://doi.org/10.1145/1414004.1414016

Yiftachel, e. a. (2011). The Study of Resource Allocation among Software Development Phases: An Economics-Based Approach. *Advances in Software Engineering*, *2011*, 1–21. https://doi.org/10.1155/2011/579292

Zelkowitz, M. V. (1978). Perspectives in Software Engineering. *ACM Computing Surveys (CSUR)*, *10*(2), 197–216. https://doi.org/10.1145/356725.356731