MITIGATING SCHEDULE OVERRUNS CAUSED BY MISINTERPRETATION

A Thesis

by

WARREN WILLIAM ROONEY

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirement for the degree of

MASTER OF SCIENCE

December 2016

Major Subject: Mechanical Engineering

ABSTRACT

Many large-scale, complex engineering systems experience significant cost and schedule overruns during their developments. There are many factors that contribute to these overruns, including increased system complexity, task rework, and the inability to exhaustively test all states and configurations of a given system, which often leads to re-design efforts. In this work, we focus specifically on task rework and its impact on project schedule overruns. We demonstrate that heavier tail phenomena present in large-scale program development duration can be caused by task rework. Within this context, we hypothesize one cause of task rework in a project development effort. We develop a computational framework for estimating the information content of a set of task instructions and the expected time to task completion. We then compare the computational results to experimental data collected for the same set of instructions to validate the model for constructing a paper airplane. This reveals that heavier tailed duration phenomena present in many large-scale project development efforts can arise due to task rework caused by misinterpretation.

To my parents for their love and support throughout the years. Without them I would not be where I am today.

ACKNOWLEDGMENTS

I would like to thank the Texas A&M University Mechanical Engineering Department for giving me the opportunity to pursue my Masters Degree. Special thanks to Douglas Allaire, Daniel Mcadams and Rachel Smallman for serving on my committee and helping me throughout the research process.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I
INTRODUCTION AND LITERATURE REVIEW[*]

Introduction

Many recent large-scale engineered system program development projects have seen substantial and unexpected cost and schedule overruns [2]. This phenomenon is indicative of heavier tailed distributions governing stochastic duration and cost of large-scale project efforts. Qualitatively, heavy tailed distributions have tails that are thicker than those of an exponentially distributed random variable. However, here, we are not referring specifically to heavy tailed distributions, only distributions whose tails are thicker than expected (e.g., thicker than a normal approximation). As noted in Refs. [2] and [3], the single biggest driver behind increasing costs for the specific case of the aircraft industry, is schedule growth. The systems involved in the aircraft industry are cyber, electrical, and mechanical in nature and are prevalent in many other industries. Thus, we focus here on schedule overruns and claim that this is a reasonable proxy for program cost as well.

Until recently, heavier tailed phenomena in stochastic project duration distributions were not modeled in standard techniques such as the Project Evaluation Research Task (PERT) method [4]. Given the recent recognition of unexpected large schedule overruns in many program developments, approaches to modeling the heavier tailed nature of stochas-

[*]Reprinted with permission from "An Information-Theoretic Model of Project Schedule Overruns Caused by Task Rework: A Case for Newspeak" by Warren Rooney and Douglas Allaire, 2016. ASME 2016 International Design Engineering Technical Conferences Computers and Information in Engineering Conference, Vol. 1, pp. 1-11, Copyright 2016 by ASME.

tic project duration distributions are now actively being studied. These efforts will likely lead to the reduction of some of the surprise associated with substantial schedule overruns. Our work here is motivated by the desire to reduce the heaviness of the tails of program development duration distributions, and hence reduce substantial schedule overruns themselves.

There are many causes cited for project schedule overruns. These include increasing product complexity, the vast number of states and configurations of a system that must be tested, task rework, procurement delays, re-design during system integration, and reliance on an outdated model of the systems engineering process [2, 5, 6, 7, 8, 9]. Our goal in this work is not to examine the impact of each of these potential causes for schedule overruns, but instead to focus on a specific aspect of task rework that we hypothesize leads to heavier tailed phenomena in project durations. Task rework refers to repeating certain tasks in a project caused by the discovery of errors in previous work, which can lead to a cycle of rework as more errors are made in correcting rework [7]. Previous work related to studying causes for task rework include studying the impacts of changes to a system during a design process and how those changes are propagated [10], studying disruptions, such as designer mistakes, in the design process by modeling them as impulses through a systems [11], studying the concept of mistake making in a theoretical sense [12], and developing a failure modes and effects analysis that includes many potential causes for rework, including misinterpretation of communication during a design process [13]. Generally speaking, in large-scale, complex system design and development efforts, there are many tasks that must be completed, and hence there are many different sets of task instructions that must be communicated between different members of the project team. We hypothesize that misinterpretation of task instructions, even with small probabilities of occurrence, can lead to heavier tailed project duration distributions. Further, we hypothesize that through more careful design of such instructions, the impacts of misinterpretation can be mitigated, and the heaviness of the tails in project duration distributions can be reduced.

The rest of the paper is organized as follows. In Chapter 1, we present background on traditional methods for stochastic schedule modeling. In Chapter 2, we formally present our hypothesis and the objectives of our work. In this Chapter we also introduce our information-theoretic approach to modeling information content in task instructions. In Chapter 3, we define our computational methodology, which focuses on a critical path model of tasks that must be completed for project success. This is followed by a presentation of the experimental setup and discussion of our results, which consist of numerical evidence of the relationship between the information content of a set of task instructions and task duration, as well as evidence of the heavier tailed behavior of task duration that results from misinterpretation. Chapter 4 provides a summary of the results and the objectives which were achieved as a result of this work. Furthermore, a discussion of our future experimental validation work is presented as part of the conclusion of this work.

Background on Stochastic Schedule Modeling

Engineering systems are becoming more complex, involving substantial numbers of components and their interactions. For such systems, the need to adequately predict system behavior with mathematical models, the need to predict the time and expense for completing project tasks, and the need to manage potentially globally distributed design teams are significant challenges. Increasingly, such large-scale complex engineering projects encounter significant cost and schedule overruns. Extreme examples are the Hubble Space Telescope's expensive repair costs [14], the V-22 Osprey tilt-rotor aircraft's substantial schedule overrun [15], as well as the recent Boeing 787 cost and schedule overruns caused by many subsystem test failures, as well as in service problems [16]. Often, the complexity of the system is blamed for such cost and schedule overruns. In previous work, we defined a metric for system complexity based on quantitatively estimating the potential of a system to exhibit unexpected behavior in its quantities of interest [9]. Here, the quantity of interest we focus on is the project duration, and we hypothesize that misinterpretation leads to task rework that leads to unexpected project delays. These delays are unexpected because rework due to misinterpretation of task instructions is not modeled in current stochastic schedule methods. In the same manner that uncertainty in inputs to computational models is mapped to uncertainty in outputs of those models, we seek to quantify the uncertainty in how information is communicated between parties in a design team, and how that impacts project duration.

The process of modeling and predicting a project's duration has been the source of research since the 1950’s [4]. The PERT method was first developed in 1959 by the United States Navy to attempt to control the development process of the Polaris Fleet Ballistic Missile Program [4]. The initial method required time estimations for every task based off estimates from experts in the respective fields, and the sequence of tasks to be completed [4, 17]. Sequencing the tasks into a specific order in the development of PERT also led to the creation of the critical path model (CPM), or the sequences of events crucial to on-time project completion [4]. As a part of the PERT methodology, parameters for the time estimation of task completion are elicited from expert opinion. The process consists of estimating “most-likely”, “optimistic”, and “pessimistic” task durations, which are then used to estimate the mean and variance of a given task’s duration. Malcolm et al. further proposed to model this information as a random variable with a beta distribution as a proxy for a normal distribution (albeit with finite support). As noted in Ref. [18], the beta distribution is similar to a normal distribution if both the kurtosis and variance is equivalent to that of the normal distribution. However, this still leaves a free parameter to be estimated for the beta distribution [17, 19].

Though it has been argued that the family of potential beta distributions resulting from this modeling process is accurate for a wide range of modal values, there are numerous studies of projects which have run significantly over time or cost [20, 16, 21, 22, 23]. Examples of such projects range from the development of Boeing’s 787 Dreamliner to large-scale information technology projects, of which one in six was found to have a cost overrun of 200% and schedule overrun of 70% [20, 16]. These numbers are indicative of heavier tailed phenomena in both project cost and project schedule overruns. One of the recurring observations from analysis into why these projects exceeded budget and schedule was the inherent uncertainty in the project estimation process [21, 22, 23, 24]. Understanding how to better detect and minimize this uncertainty in project management has been an ongoing area of research (see, e.g., Refs. [24, 25, 26, 27]). In regards to the PERT method, Hill et al. note that experts' estimates of task duration is most underestimated when the task is either unclear or complex [25]. Thus, more robust methods of time and cost estimation were developed for use in the PERT methodology, including the use of distributions with heavier tails to account for extreme events that were not being modeled properly by the original beta distributions of the method [18, 28, 29]. Recent emphasis on heavy tailed distributions in state-of-the-art stochastic modeling of project schedule durations is suggestive that data collected on recent project schedule overruns is being attributed in part to the presence of heavier tails in the distribution of project duration. If recent project durations are indeed accurately modeled with heavier tailed distributions, then a critical task is the identification of causes of these extreme events that are much more probable than once expected. Further, the identification of how to mitigate the effects of such causes is also of great interest. In the following Chapter we hypothesize one such cause whose effects have the potential to be mitigated. That cause is misinterpretation of task instructions.

A Note On Heavy Tails

While some of the literature surveyed suggests that schedule overruns of large-scale engineered system development projects are indicative of heavy tailed phenomena in their distributions, we have been careful not to claim these distributions are truly heavy tailed in a mathematical sense. Instead, we have referred to the "heaviness" of the tails, by which we mean in relation to normal assumptions. Indeed, by definition, the distribution of a random variable, X, has a light right tail if $\mathbb{P}(\{X > 0\}) > 0$ and $X \leq C$ for some constant C [30]. Historically, no project has an infinite amount of time to complete, and therefore, distributions of project durations are likely light tailed (just heavier than once thought in the traditional PERT method).

Numerical study of the tails of task duration distributions estimated by our methodology, which incorporates task rework, are not heavy in the sense that eventually, the tail of an exponential random variable is thicker than the tail of our empirically derived distribution. Under certain assumptions, our model of schedule duration impacted by task rework is a compound geometric random variable. Such distributions have received careful study in insurance mathematics and reliability analysis [31, 32, 33]. It is a topic of future work to compare historical project duration data with the tails of analytic compound geometric distributions. Such a study could provide significant evidence of the impact of rework on duration, a clearer picture of how assumptions involving where and when task rework takes place in the development process, and a quantitative framework for studying how best to mitigate the impacts of rework (e.g., via tailored instruction design processes).

CHAPTER II

RESEARCH METHODOLOGY[*]

Hypothesis and Objectives

For large-scale, complex engineered systems, there are a great deal of tasks that must be completed during development efforts. Instructions for completing these tasks must be communicated between parties working within the project team. However, it is the case that misinterpretation of such communications can occur. In fact, in a large-scale complex system development effort, with potentially many thousands of pieces of information that must to be communicated, we claim that it is likely that misinterpretation will occur at some point in the process. When misinterpretation goes unnoticed, the result can be necessary task rework. A cause of such a situation was eloquently stated by Robert McCloskey when he famously stated, “I know you believe you understand what you thought I said, but I’m not sure you realize that what you heard is not what I meant” [34]. If an individual responsible for a particular task in the critical path of a project believes they understand what they have been asked to do, but in fact have misinterpreted the instructions, then the errors this misinterpretation causes will only be discovered later in the development process (if at all) and lead to rework. Thus, we pose the following hypothesis:

Hypothesis: Task instructions in large-scale engineered system program development ef-

[*]Reprinted with permission from ”An Information-Theoretic Model of Project Schedule Overruns Caused by Task Rework: A Case for Newspeak” by Warren Rooney and Douglas Allaire, 2016. ASME 2016 International Design Engineering Technical Conferences Computers and Information in Engineering Conference, Vol. 1, pp. 1-11, Copyright 2016 by ASME.

forts can be misinterpreted. This misinterpretation can go undetected for a period of time until errors it has caused are discovered. This occurrence leads to costly iterations in the development process that can lead to heavier tailed phenomena in project duration distribution estimates.

Here, we accept the first part of the hypothesis as fact and further assume the second part of the hypothesis to be true. Our goal then is to present evidence of the third part of the hypothesis: that misinterpretation can lead to heavier tailed phenomena in project duration distribution estimates. Our specific objectives in this work are then: 1) to develop a computational framework to provide numerical evidence of the validity of our hypothesis; 2) to demonstrate that misinterpretation can lead to a heavier tailed distribution for project duration; and 3) to design an experiment to validate the results of the computational framework, and consequently demonstrate that we can reduce the heaviness of the tail of project duration distribution estimates by improving the information content of task instructions.

In this work, to quantify the potential for misinterpretation of a given task instruction, we estimate the information entropy of a discrete random variable, Y, where Y can take values $y_1, y_2, \ldots, y_n$ with positive probabilities respectively. Here, the values Y can take represent different states of the task after the instruction is followed (e.g., successfully completed, failed, unsuccessfully completed but propagated) that result from the interpretation of the instruction for the task. The information entropy of the random variable, Y, is then defined as

$$H(Y) = -\sum_{s=1}^{n} p(y_s) \log p(y_s). \tag{II.1}$$

The entropy of a random variable is a measure of the uncertainty of the random variable and is essentially a measure of the amount of information required to describe the random variable completely [35]. Thus, information entropy is precisely what is required to quantify the misinterpretability of a given instruction or set of instructions. That is, the higher the information entropy, the more misinterpretable an instruction set, and vice versa. Our goals in the context of information entropy of a set of instructions are then to create a computational framework for estimating the information entropy of a set of instructions given probabilities of misinterpretation and to demonstrate the trend of increasing project duration with increasing instruction entropy. The use of information entropy in design has been incorporated in many previous works. For example, Refs. [36] and [37] discuss the concept of using information theoretic quantities as design metrics for concepts such as design freedom and system complexity. Refs. [38] and [39] develop information theoretic approaches to design based on probabilistic sensitivity analyses and optimization in the context of design process dynamics. Our approach thus builds on such successes with using information theoretic methods and applies the concept to misinterpretation during a design process.

Computational Framework Methodology

To provide evidence of the validity of our hypothesis, we develop a computational framework for numerically analyzing the impact of misinterpretation on project duration. The approach is to simulate the interpretation of a set of instructions for completing tasks on the critical path of a project. Thus, when a misinterpretation occurs, rework must occur at some point on the critical path, which we have hypothesized can lead to larger tails in project duration distributions. Additionally, by focusing on the critical path the need to worry about parallel processes in a design process is suspended. In this section the computational framework is presented as well as a discussion on the scope of the approach taken.

Consider a set of instructions for completing tasks on the critical path of a project. To rigorously quantify the potential for misinterpretation, the information entropy of the distribution of possible end states that can result from following the instructions (via any given interpretation) is sought. For example, if there is only one instruction that has two interpretations, then these interpretations can be labeled y_1 and y_2, with the assumption that each interpretation leads to a different end state. Then, as described in Chapter 2, the information entropy of the instruction is given as $H = \sum_{s=1}^{2} p(y_s) \log p(y_s)$. If there are subsequent instructions then different possible interpretations of each instruction will lead to a branching structure of potential states. For example, consider two serial instructions that need to be followed to complete a design task, each with three potential interpretations. This concept is shown notionally in Fig. 2.1, where each branch represents on of the three possible interpretations. Thus, at each node in the tree, an interpretation of an instruction must be arrived at and followed. The probability of a given interpretation is denoted on the figure as:

$$\mathbb{P}(\{<i>=<\alpha> \mid <i-1>=<\beta>\}), \tag{II.2}$$

where i is the instruction number, α is the interpretation of the i-th instruction, and β is the interpretation of the $(i-1)$-th instruction.

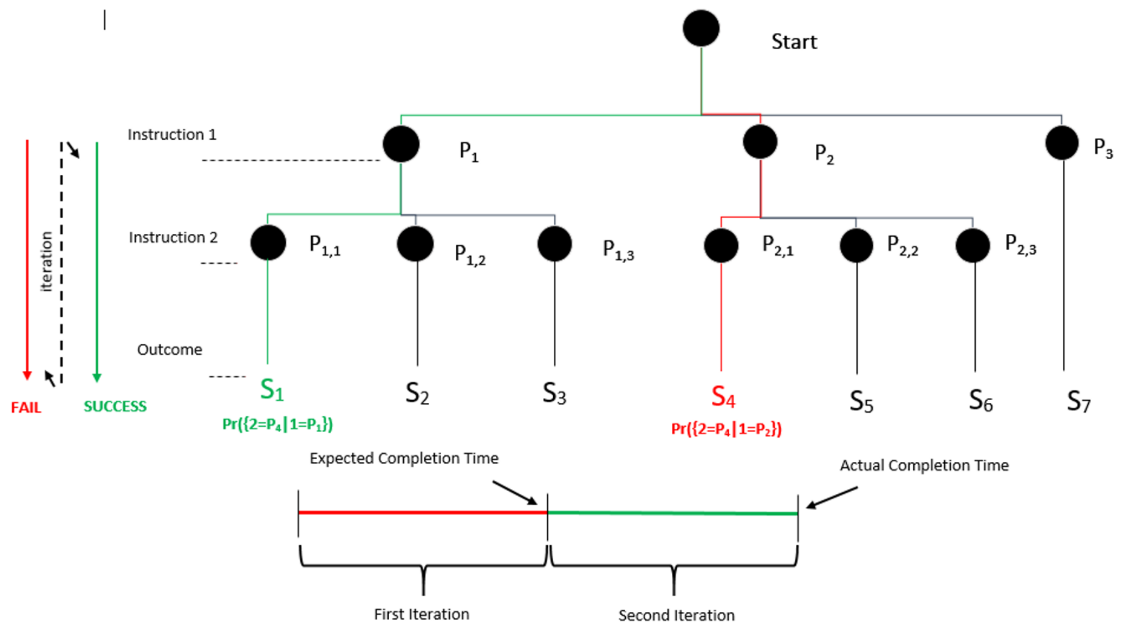

Figure 2.1: Representation of seven different outcomes from a set of two instructions in series with three interpretations each.

In this case, there are seven different possible interpretations of the series of instructions that lead to seven different end states which are denoted as $S_i, i = 1, 2, ..., 7$. Fig. 1 also illustrates an important assumption which was made to assist with simplifying the computational model. This assumption was to group the almost unlimited number of possible interpretations for an instruction into three general categories:

1) Correct.

2) Incorrect, but plausible.

3) Grossly incorrect such that the next instruction does not make sense.

This third category is the reason for the path denoted P_3 to proceed directly to state S_7 since the second instruction would not make sense and the design process would cease at that point. Clearly, some of the potential end states will be very unlikely. In fact, it is likely the case, that even with potential for misinterpretation, the correct interpretation of each instruction is considerably more probable than any other case. However, as is demonstrated in Chapter 3, any potential for misinterpretation dramatically changes what should be anticipated in terms of project duration. For a given set of instructions, interpretations, and probabilities for each interpretation, the different paths to each possible end state can be enumerated and the information entropy for an instruction set can be computed. This entropy, for a given instruction set, is then a measure of the amount of information needed to ensure there is no misinterpretation of the instructions. Viewed in this manner, it is also a metric that can be used to identify more informative instruction sets, which as demonstrated in Chapter 3, can lead to reduced project durations.

The specific numerical simulation that was developed to provide evidence of the hypothesis involves a series of instructions. Each instruction has three possible interpretations in the same manner as that shown in Fig. 1, where again, only two instructions are shown. In this figure, each left hand branch, denoted P_1, $P_{1,1}$, or $P_{2,1}$, is considered the correct interpretation. A potential path through the instructions is shown with red branches, where the first instruction was interpreted incorrectly but plausibly, and the second instruction was interpreted correctly. This leads to the end state denoted S_4, that is a failed completion of the instruction set. A rework iteration is then required. Another assumption of computational model is that all rework begins at first instruction. A discussion of this, and the other assumptions is included in more detail below when the computational model algorithm is introduced. The correct interpretations are achieved in this rework iteration, as shown by the green branches that arrive at state, S_1. The task duration is then the sum of the duration of each iteration through the instructions. The duration of each instruction is defined as a beta random variable, which is denoted $B_i \sim Beta(\alpha_i, \beta_i)$ where i is the instruction index, α=2 and β=4 [28]. Each instruction has an upper and lower time bound, and as such the duration of each iteration is written as

$$X_k = \sum_{i=1}^{N} lb_i + (ub_i - lb_i) * B_i \tag{II.3}$$

where k is the index of the iteration number, N is the number of instructions, lb_i is the lower bound of each instruction, and ub_i is the upper bound of each instruction. The duration of the project is then written as

$$T = \sum_{k=1}^{k} X_k \tag{II.4}$$

Where k denotes the number of iterations required for successful completion of the instruction set. Since the number of iterations depends on the entropy of the instruction set and is a discrete number, k is also a randomly distributed variable which results in T having a compound geometric distribution [40]. Because of rework, T is generally larger than the expected duration of the project as estimated by traditional means, where the duration of each instruction completion in the project is summed to arrive at the distribution of the project as a whole. The beta distribution is used to represent the distribution of each instruction completion time in the project as described by the traditional beta distribution found in PERT. The PERT beta distributions were used with an assumption that the instructions were estimated accurately to within ±10%. This assumption allows the use of PERT beta distributions as opposed to other more "modern" distributions because an accurate estimation of PERT time parameters to within ±10% form the bounds in which many "modern" distributions fall as shown in Fig. 2.2 [1].

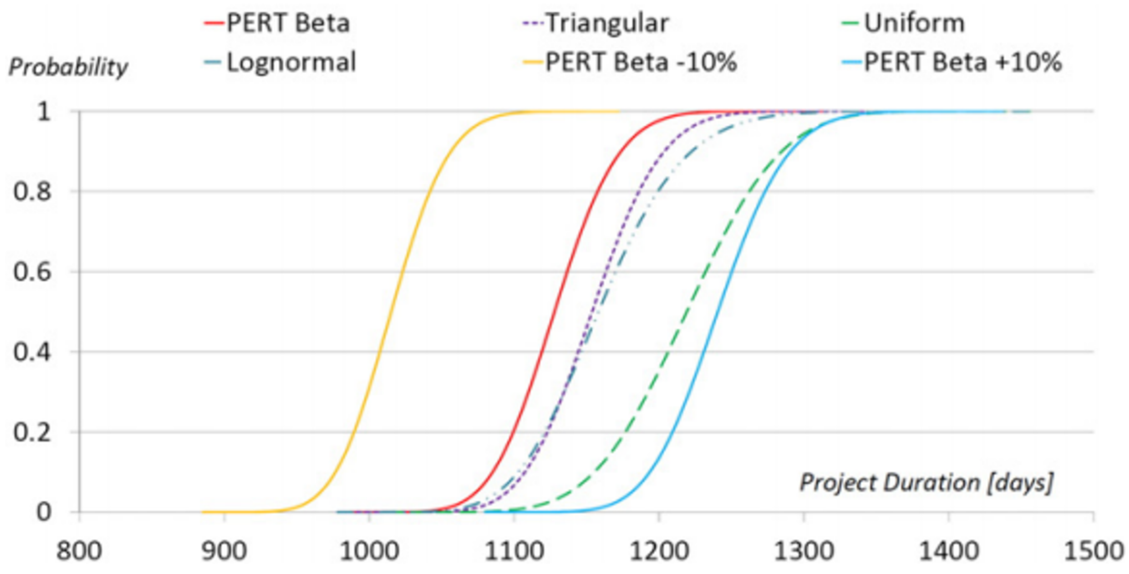

Figure 2.2: Figure adapted from [1]. Note how the PERT Beta ±10% distributions form an upper and lower bound which the other distributions fall between.

For the results presented in Chapter 3, there are several assumptions made about the potential path through a set of instructions to arrive at an end state. These assumptions simplify the interpretation of the results, but their relaxation will be required for the computational framework to more accurately reflect reality if the framework is to be used for actual analysis of project durations impacted by potential misinterpretation of instructions. Relaxing these assumptions is a topic of future work. The assumptions here assume a memoryless property for every instruction, so that each interpretation for a given instruction does not depend on the previous instructions' interpretation. This assumption holds within a given iteration, as well as during rework iterations. Mathematically, it follows that

$$\mathbb{P}(\{<i>=<\alpha> \mid <i-1>=<\beta>\}) = \mathbb{P}(\{<i>=<\alpha>\}), \qquad \text{(II.5)}$$

where the notation is defined as before. This assumption implies that no learning happens during the rework process and that every instruction has the same number of possible interpretations with fixed probabilities, regardless of the current state of the process. Clearly, certain interpretations would impact subsequent instruction interpretations and even indicate in many cases that a mistake has been made prior to arriving at the end of the instruction set. We claim that the general implications of using our framework discussed in Chapter 3, are not affected by these assumptions. Addressing these types of assumptions is necessary for a given specific set of instructions for a given project, but the general trends and tail behavior of the duration distribution will not be affected significantly. A detailed analysis of this claim is a topic for future work. Another assumption present in the results is that the duration of following each instruction is an independent beta distribution, though not necessarily identical. We also assume that a full iteration takes place once the end state is arrived at, even if a misinterpretation has occurred. If a misinterpretation does occur, the location of the first misinterpretation for that iteration is identified along with the correct interpretation. For example, if in the first iteration, instructions 1,3, and 4 are misinterpreted only the first instruction will be identified as incorrectly interpreted for the second iteration. This will change the probabilities for instruction 1, but the remaining 4 instruction probabilities will stay the same in accordance with the previous assumption. This assumes that the person in charge was absent during the design process, but is familiar enough with the task to identify mistakes from final results.

This section is concluded by presenting the algorithm for computing the entropy of a given instruction set in Algorithm 1. The algorithm is a Monte Carlo based approach to estimating the probability of ending at each possible end state of a given instruction set after one pass through the instructions.

Algorithm 1: Instruction Entropy Estimation.

```
    Number of instructions, $N$, Probabilities of correct instruction interpretations, $p_i$,
$i = 1,2,\dots,N$, Number of samples, $M$. Estimate of the instruction set information entropy,
$H$.
    Initialize: $S_s = 0$ for $s = 1,2,\dots,2^{N+1}-1$.
for $j = 1:M$ do
    Set all $S_s$ equal to 0
  for $k = 1:5$ do
    Sample $u_k$ from $U[0,1]$ distribution
    Compare $u_k$ to $p_i$ where $i = k^{th}$ fifth of $N$
    (i.e. $k = 1, i = 1,2,3$)
    **if** $u_i \leq p_{i_{(1)}}$ **then**
  Set $S_s$ to $S_s + 1$ for the correct interpretation
      **if** $u_i > p_{i_{(1)}}$ or $u_i \leq p_{i_{(1)}} + p_{i_{(2)}}$ **then**
  Set $S_s$ to $S_{s+1}$ for the incorrect, but plausible interpretation
      **else**
  Set $S_s$ to $S_{s+1}$ for the gross misinterpretation interpretation
  Set $P_s = S_s/M$ for $s = 1, 2, \ldots, 2^{N+1} - 1$.
Set $H = -\sum_{s=1}^{2^{N+1}-1} P_s \log(P_s)$.
```

From this process, another algorithm, Algorithm 2, is presented to demonstrate how the number of iterations and corresponding time to completion for an instruction set was modeled in the computational framework.

Algorithm 2: Time to Completion Estimation.

Number of instructions, N, Probabilities of correct instruction interpretations, p_i, $i = 1, 2, \ldots, N$, Number of samples, M, Upper and Lower Time Bounds of Each Instruction, UB and LB. Estimate of the time to completion of the instruction set, T.

Initialize: $itr_j = 0$ for $j = 1, 2, \ldots, N$.

```
for $j = 1 : M$ do
    Set all $itr_j$ equal to 0
  for $k = 1 : 5$ do
    Sample $u_k$ from $U[0,1]$ distribution
    Compare $u_k$ to $p_i$ where $i = k^{th}$ fifth of $N$
    (i.e. $k = 1, i = 1,2,3$)
      if $u_i \leq p_{i(1)}$ then
    Set $itr_j$ to $itr_j + 1$ Exit the loop
      else
    Set $itr_j$ to $itr_j + 1$ Set first $u_i > p_{i(1)}$ to 1 Rerun through loop
  for $k = 1 : 5$ do
    Convert $itr_k$ to beta random time using Eq. 2.3
    Set $T = \sum_{i=1}^{N} X_i$.
```

The algorithms assume that the upper and lower time bounds as well as the probabilities of each instruction are known. The method of obtaining these is explained in detail in the following section, and while the assumptions are not representative of a realistic scenario, they are necessary to validate the computational model to experimental results. It is a topic of future work to perform instruction set experiments without prior knowledge of instruction probabilities.

Experimental Setup and Methodology

The computational framework was designed to achieve research objectives 1 and 2. However, to achieve research objective 3, and validate the results from the computational model, an experiment was developed to measure the impact of rework on project schedules. The experiment designed was for paper airplane construction based off the experiment designed by Dr. Steven Eppinger from MIT [41]. To measure the impact of rework on the experiment, a set of instructions with varying degrees of clarity were created. The instruction set developed in this experiment consisted solely of text, and included five instructions for building a paper airplane which simulated the critical path of a project. The final instruction set which was used for the experiments can be found in Appendix A. The paper which was used to create the paper airplanes can be found in Appendix B. Participants for the experimental portion of the study were taken from the Mechanical Engineering Department at Texas A&M University in order to simulate engineers performing some standardized task. Due to the simplistic nature of the experiment as well as the level of intelligence of the participants, unfamiliarity with paper airplane construction was assumed to be a negligible factor in time to completion.

To validate the hypothesis, the experimental model needed to be able to recreate the results from the computational model. In order to accomplish this task, the experimental portion of the research was broken into two categories which will be discussed in more depth in the following sections: pilot tests, and the actual experiments. The results from the experimental study were then analyzed using the complimentary cumulative distribution function (CCDF) from the experimental times to completion. This CCDF was then compared to the CCDF from the computational times to completion in order to validate statistically whether the two CCDF's come from the same distribution. In order to prove whether the CCDF's matched or not, a Kolmogorov-Smirnov (KS) test was chosen to statistically refute the null hypothesis, H_o, or claim it was plausible. The null hypothesis in this research was that the two distributions, experimental and computational, came from the same distribution. This resulted in a research hypothesis, H_1, which was that the two results did not come from the same distribution. The p-value of the KS test was ultimately the statistic used to either support or refute the hypothesis introduced for this research.

Pilot Tests

As mentioned in the computational framework section, the Monte Carlo simulation used the assumption that the probabilities of each instruction, as well as the upper and lower time bounds for completion of each instruction were known. In order to gather this information for the specific instruction set in Appendix A, a pilot test was created. This test was designed to have a small group of 5-10 participants iterate through the instruction set one instruction at a time. The participants were responsible for recording the individual time to completion for each instruction, while the researcher was responsible for recording the interpretations for each instruction per participant. To ensure that participants were not influenced by the decisions of other participants, dividers were used to keep their work individual. Once the participants had gone through the entire instruction set, the pilot study was run a second time to get a new value of times to completion. This was done to account for the familiarity a participant would have with the instruction set in the real experiment if they had to iterate through the process again, and the corresponding faster times to completion for each instruction. With the results from the pilot study, the final instruction set could be simulated in the computational framework to obtain a CCDF of the estimated time to completion of the instruction set. The results of the pilot tests will be discussed further in the next chapter.

Actual Experiment

The actual experiment was designed for a larger group of 50-100 participants. Participants would iterate through the instruction set from start to finish, instead of performing one instruction at a time as in the pilot studies. Each participant was responsible for timing themselves for each instruction as well as the cumulative process for each iteration, and time would stop once the participant reached the end of the instruction set or a gross misinterpretation occurred as defined in the Computational Methodology section. If there was a misinterpretation of one or more of the instructions, the researcher would identify to the participant the earliest instruction misinterpreted, and the participant would begin from the beginning with the new information. This process was to align the experiment as closely as possible to the assumptions in the computational model, and would continue until the participant correctly interpreted every instruction and arrived at the desired final end state. It should be evident that not all assumptions in the computational model could be matched exactly, such as the memoryless process, however an auxiliary function of the experiment was to validate the assumptions for the computational model in this specific experiment. Once the time data and number of iterations for each participant was collected, an empirical cumulative distribution function (ECDF) as well as the corresponding CCDF could be created for the data set and compared with the CCDF of the computational simulation. The results of the computational framework, experimental tests, and comparisons of the two distributions will be discussed in the following chapter.

CHAPTER III

RESULTS AND DISCUSSION[*]

Computational Simulation

In this section, the computational framework described in in the previous chapter was used to conduct numerical experiments to provide evidence of the validity of the hypothesis. As previously mentioned, there are many assumptions which limit the complexity of the computational framework, however, this research claims that the trends emphasized are insensitive to these assumptions.

The second objective of the work is to demonstrate that misinterpretation can lead to heavier tailed phenomena in project duration distribution estimates. For this, three numerical experiments using the framework consisting of five instructions, each with the same probability of correct interpretation, were conducted. The number of trials for this experiment was 1,000,000. In these three experiments, the path representing the gross misinterpretation (i.e. P_3 in Fig. 1) were ignored. The heavier tailed phenomena is demonstrated by the CCDF, also known was the tail distribution, $\overline{F}_T(t) = 1 - F_T(t)$, where $F_T(t)$ is the empirical cumulative distribution function of the project duration estimated by the computational framework, with that of a traditional PERT beta approximation to project duration, which is denoted as $\overline{F}_{\mathscr{B}}(t) = 1 - F_{\mathscr{B}}(t)$. The results of this experiment for correct

[*]Reprinted with permission from ”An Information-Theoretic Model of Project Schedule Overruns Caused by Task Rework: A Case for Newspeak” by Warren Rooney and Douglas Allaire, 2016. ASME 2016 International Design Engineering Technical Conferences Computers and Information in Engineering Conference, Vol. 1, pp. 1-11, Copyright 2016 by ASME.

interpretation probabilities of each instruction fixed to 0.999 are shown in Fig. 3.1.

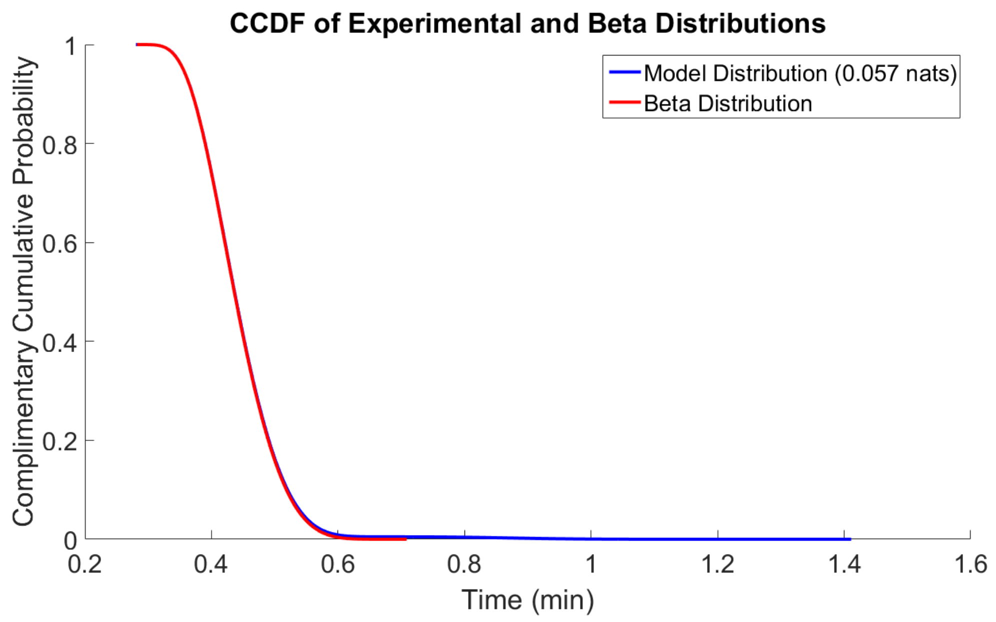

Figure 3.1: The CCDF's of a set of instructions as estimated by a traditional PERT Beta estimation of instruction completion duration (red) and this algorithm that has a 0.1% chance of misinterpretation rework (blue).

The y-axis in this figure represents the complimentary cumulative probabilities, and the x-axis is the estimated time to completion in minutes. The entropy value denoted in nats for the experimental distribution describes the entropy of this instruction set as denoted in Equation 2.1. An informal pilot test was conducted to obtain rough estimates for the upper and lower time bounds of each instruction to conduct this numerical experiment. As such, the quantitative value for the estimated time to completion is inaccurate and not the critical element of this figure. Rather, the qualitative difference of the two distributions is the element worth noting. The two distributions are very similar since the chance of misinterpretation for this first study was limited to 0.1% per instruction, however the distribution containing the misinterpretation has a significantly heavier tail than the distribution formed from the traditional PERT beta approximation. Although the probability of tail area events beyond 0.7 minutes is extremely small, the computational distribution still provides evidence that a project can exceed the most pessimistic estimated time to completion described by the PERT beta approach by as much as 100%.

The second numerical experiment followed a similar approach to that of the first, except this time the chance of misinterpretation was increased to 1.0% in each instruction resulting in a probability of correct interpretation of 0.99. Again, the paths which represented the gross misinterpretations were ignored for this study, and the results are shown in Fig. 3.2.

Again, the qualitative differences between the two distributions is worth analyzing more than the actual estimated time to completions for this study. First, the entropy of this experimental distribution has increased from 0.057 nats in the first experiment to 0.404 nats in the second experiment, which is to be expected since the chance of misinterpretation has increased. Second, the two distributions start to diverge from each other much more than the distributions in the first experiment did. Here, not only does the experimental distribution have an even heavier tail than in the first experiment, but the probability of a project exceeding the upper limit approximated by the PERT beta approach is almost 10 times higher than the first experiment. To validate this trend, a third numerical experiment was run similar to the first two with the chance of misinterpretation increased to 10.0%

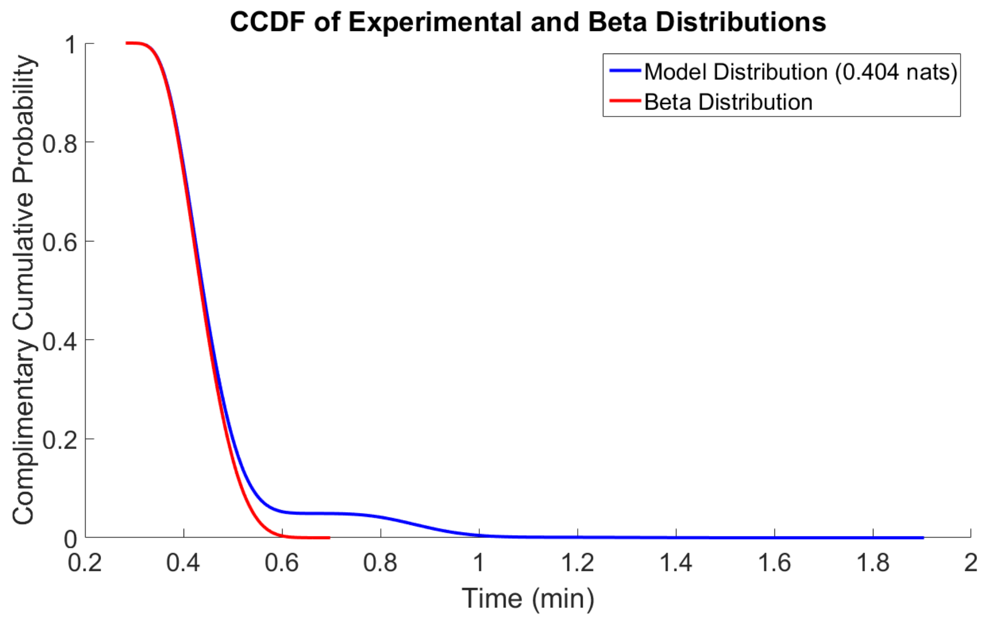

Figure 3.2: The CCDF's of a set of instructions as estimated by a traditional PERT Beta estimation of instruction completion duration (red) and the algorithm that has a 1.0% chance of misinterpretation rework (blue).

while neglecting the paths denoting the gross misinterpretation. The results of the third numerical experiment are shown in Fig. 3.3.

From the third numerical experiment, it is evident that as the entropy increases from 0.057 to 0.404 to 2.345 nats, so too does the heaviness of the computational model tail. Another interesting feature is that as the probabilities become more extreme (i.e. 99% to 90%), the CCDF of the computational model appears to have a "stair-stepping" component of its distribution. This arises due to the compound geometric distribution of the total time to completion. As the probabilities decreased, the average number of iterations required before successful completion rose as well. However, since the number of iterations was

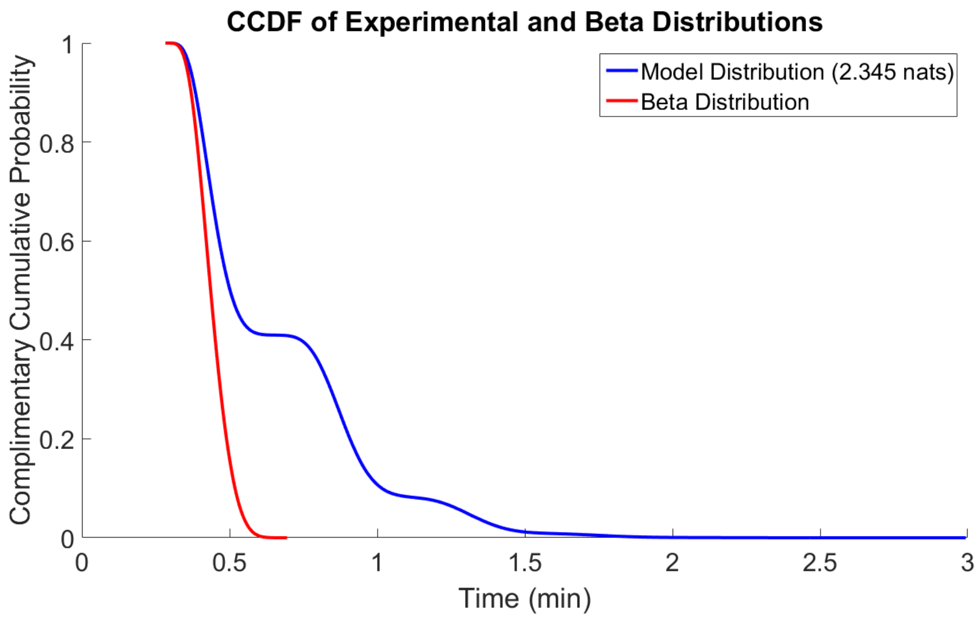

Figure 3.3: The CCDF's of a set of instructions as estimated by a traditional PERT Beta approximation of instruction completion duration (red) and the algorithm that includes a 10.0% chance of misinterpretation rework (blue).

still random, the CCDF has several changes in concavity, each of which corresponds to an increased number of iterations required.

The previous three numerical experiments provided some evidence of the trend between increasing entropy and the corresponding distribution of estimated times to completion. However, each experiment neglected the paths denoting gross misinterpretation, and focused solely on the first two paths. Thus, a fourth numerical experiment was run which included five different instruction sets of random entropy as well as the standard PERT beta distribution. The figure is shown in Fig. 3.4.

The results from the fourth numerical study demonstrate the same trends which were found in the first three studies. As the amount of misinterpretation present in an instruc-

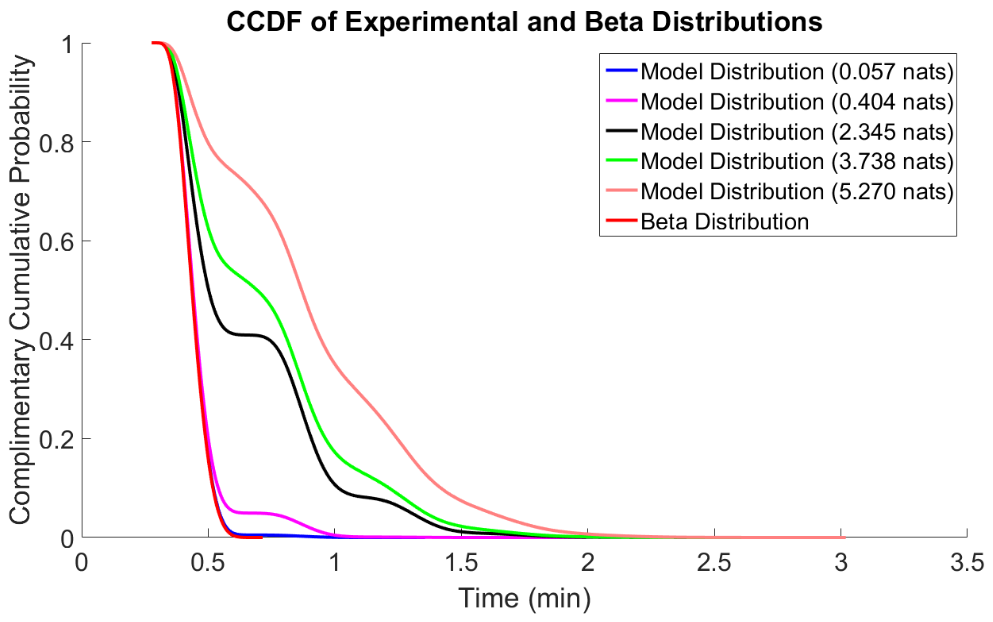

Figure 3.4: The CCDF's of estimated times to completion for 5 different instruction sets with varying entropies and the CCDF as estimated by a traditional PERT Beta approximation.

tion set increased, the distribution for estimated time to completion contained increasingly heavier tails. Additionally, the changes in concavity due to the increasing average number of iterations required, which is a compound geometric random variable, are evident in the instruction sets containing the higher entropy values. Overall, the results from the computational model appear to validate the hypothesis that an increased chance of rework causes iterations which lead to an increase in schedule duration beyond that modeled by the traditional PERT beta model. Also, the computational model and its results accomplish research objective 1, and partially achieve objective 2. However, to validate the results of the model, and completely achieve objective 2, the results of the experimental data must be compared to that of the computational model. The result of comparing the experimental data with the predicted results of the computational model is what ultimately validates or refutes the research hypothesis, and these results are discussed in the following section.

Experimental Study

As previously mentioned, the experimental portion of the research was broken up into two categories: the pilot tests, and the actual experiments. The division was to help align the experimental results with as many of the model assumptions as possible. Participants for the study were recruited from the Mechanical Engineering undergraduate student body at Texas A&M University. Since the participation was voluntary, and uncontrollable by the researchers, the sample size for the experiment was obtained haphazardly in the statistical sense, meaning the researchers could not control how participants were obtained. To accommodate for the uncontrollable haphazardness of the volunteering process, the participants were randomly assigned groups, and these groups were randomly assigned to participate in one of the two categories of the experiment. This process was to achieve experimental data which was statistically random and as unbiased as possible. The specific assumptions each category of the experiment represented and their corresponding results will be discussed in the following two sections.

Pilot Test Results

The pilot tests were necessary in order to obtain values for the interpretation probabilities of the experiment instructions shown in Appendix A. Additionally, data for an estimated upper and lower time bounds for each instruction was obtained in this portion of the experiment. These values were necessary because the computational model assumes that the instruction probabilities, and the estimated time bounds for each instruction are known. These assumptions are reasonable in the context of PERT project scheduling since experts in the field of study would be able to provide estimated times to completion for each instruction, and it was assumed the estimates would be accurate in accordance with [1].

For the pilot tests, 10 participants were used to estimate the probabilities and time bounds for each instruction. As discussed in the previous chapter, the participants were given the instruction set, and instructed to complete one instruction at a time. The participants timed how long it took to complete each instruction, and the researcher logged how the instruction was interpreted. The participants performed this process twice, to obtain more accurate time estimates if multiple iterations were required. The first instruction in particular was predicted to have a much higher estimated time to completion for the first iteration than any subsequent iterations because the participants would spend extra time on the first iteration deciding how best to interpret the instruction set. In order to keep the probability results as unbiased as possible in the second iteration of the experiment, the researcher gave no feedback on the participants' performance in the first iteration. The probabilities for the instruction set were determined using a frequentist approach, and Tables 3.1 and 3.2 summarize the results from the pilot studies. Note, in Table 3.2 $UB =$ Upper Bound Time Limit and $LB =$ Lower Bound Time Limit for each instruction.

From Tables 3.1 and 3.2, it can be seen that the second instruction and the first iter-

Table 3.1: Probability Results from Pilot Tests

Instruction 1			Instruction 2			Instruction 3			Instruction 4			Instruction 5		
P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}
0.875	0.125	0.0	0.6875	0.1875	0.125	0.8125	0.125	0.0625	0.8125	0.1875	0.0	0.8125	0.1875	0.0

Table 3.2: Estimated Time Bounds (sec) from Pilot Tests

	Instruction 1		Instruction 2		Instruction 3		Instruction 4		Instruction 5	
	UB_1	LB_1	UB_2	LB_2	UB_3	LB_3	UB_4	LB_4	UB_5	LB_5
Iteration 1	34.00	12.32	25.55	11.16	32.16	14.65	17.30	6.10	23.00	10.15
Iteration 2	24.36	6.93	23.08	9.57	22.62	10.06	28.18	1.61	25.95	9.50

ation should have the most significant impact on the time to completion since this instruction has one of the largest range of possible values, and the lowest probability of correct interpretation. However, since the sample size of the pilot tests was small due to lack of participants, the results from Tables 3.1 and 3.2 most likely underestimate the probabilities of each instruction. Another characteristic of the pilot study of interest is the distribution of the time values for the 10 participants since one of the key assumptions of the model was that the time was PERT beta distributed. Thus, histograms for the two iteration models were created and are shown in Figs. 3.5 and 3.6

Figs. 3.5 and 3.6 illustrate that while the data for some of the instruction time distributions resemble the PERT beta model with a light left tail and heavier right tail, others do not. This result was most likely caused by the small sample size of the pilot studies. However, the histograms show that the time bounds gathered from the pilot study may not be accurate enough to match the model distribution with the actual experimental one, a result which will be discussed in the following section. Nonetheless, the predicted time to

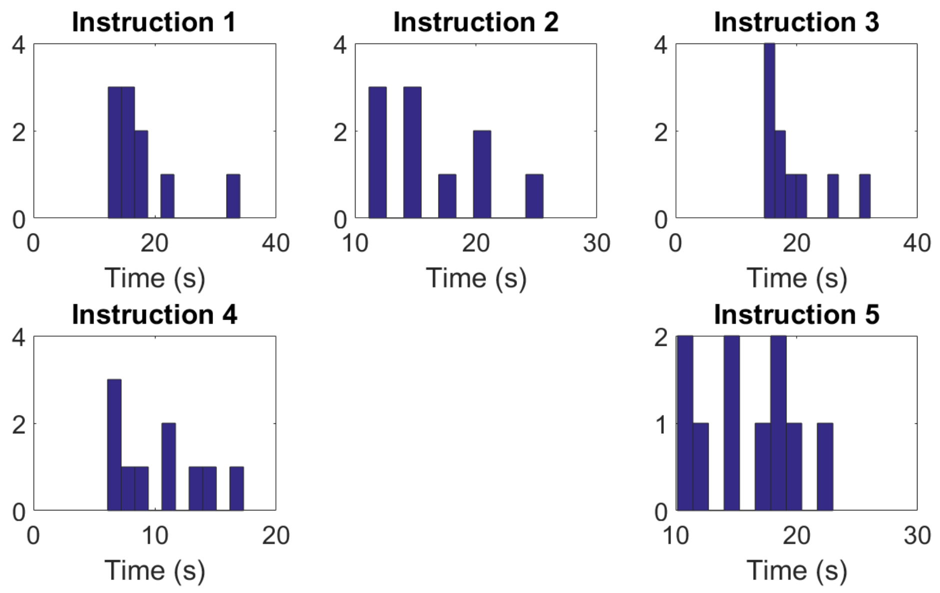

Figure 3.5: Histograms for the time distribution for the 1st iteration of the 5 different instructions with sample size 10.

completion distribution for the paper airplane experiment could be created from the data, and the result is shown in Fig. 3.7.

Fig. 3.7 illustrates that the entropy of the experimental instruction set was predicted to be 4.004 nats. The similar features which were found in all of the numerical experiments are also seen in the results from the CCDF of the pilot study. The compound geometric component of the number of iterations can be seen, and from the number of changes in concavity it would appear the average number of iterations required to successfully reach the required end state is around 2 or 3. Additionally, since the time bounds were assumed to have been accurately estimated, the tail distributions have quantitative meaning instead of just qualitative meaning as was the case in the numerical experiments. From the above figure it can be seen that the pessimistic PERT model predicts an estimated time to completion

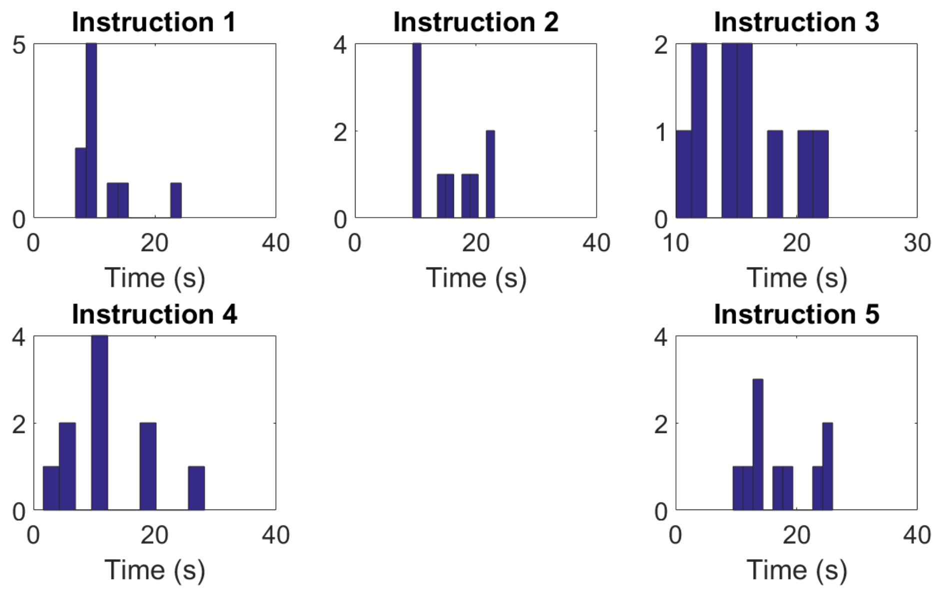

Figure 3.6: Histograms for the time distribution for any additional iterations of the 5 different instructions with sample size 10.

of approximately 1.75 minutes. However, when misinterpretation is taken into account, as is the case with the computational model, the estimated time to completion of 1.75 minutes has approximately an 80% of occurring. Additionally, times to completion between 4-6 minutes in the experiment would not be unusual as the tail of the model CCDF extends out to 7.50 minutes. Again, the small sample size of the pilot studies could result in more extreme results for probability and time estimations, however these results can and were checked with the results of the actual experiment results. The experimental results as well as the comparison between the predicted times to completion from the model compared with the actual results is discussed in the following section.

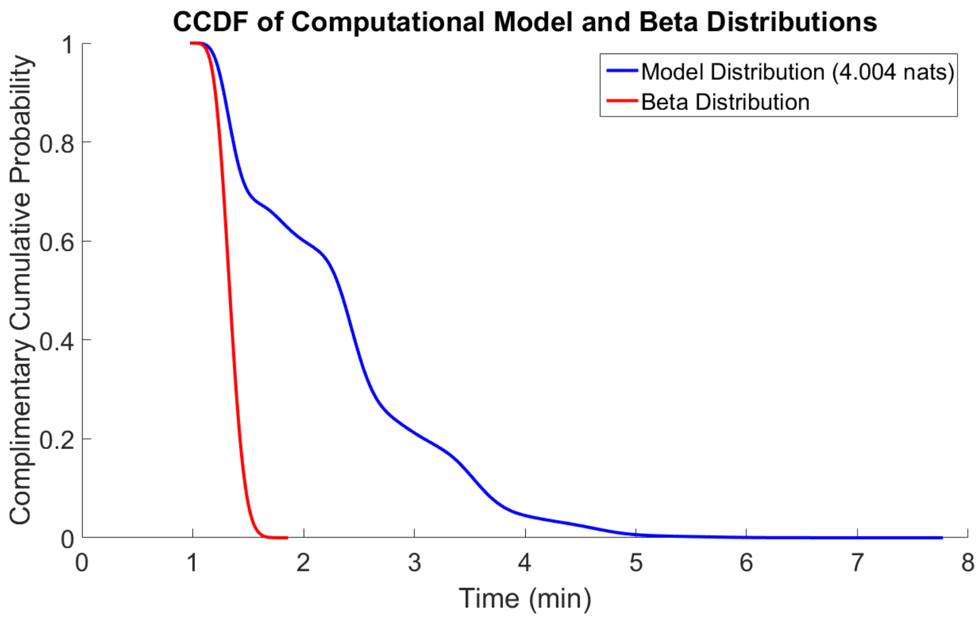

Figure 3.7: The CCDF of estimated time to completion for experimental instruction set (blue) and the CCDF as estimated by a traditional PERT Beta approximation (red) with sample size 50.

Actual Experiment Results

Once the pilot studies were completed, the remaining participants conducted the actual experiment. Data from a total of 50 participants was collected for the experimental study, which followed the format discussed in the previous chapter. The total number of iterations completed before successful completion of the experiment, and time for each iteration was collected for each participant. This data was then entered into the computer, and an empirical CCDF (ECCDF) was created for the experimental data. This ECCDF was then compared against the CCDF for both the standard PERT beta distribution, and the predicted model distribution. To determine whether the two models were from the same distribution, the CCDF's were compared using a KS test in MATLAB. The null hypothesis for the KS test was that the two models came from the same distribution, and the probability for a Type 1 error was set at $\alpha = 0.05$. Figure 3.8 shows the result from the ECCDF compared with the standard PERT beta CCDF.

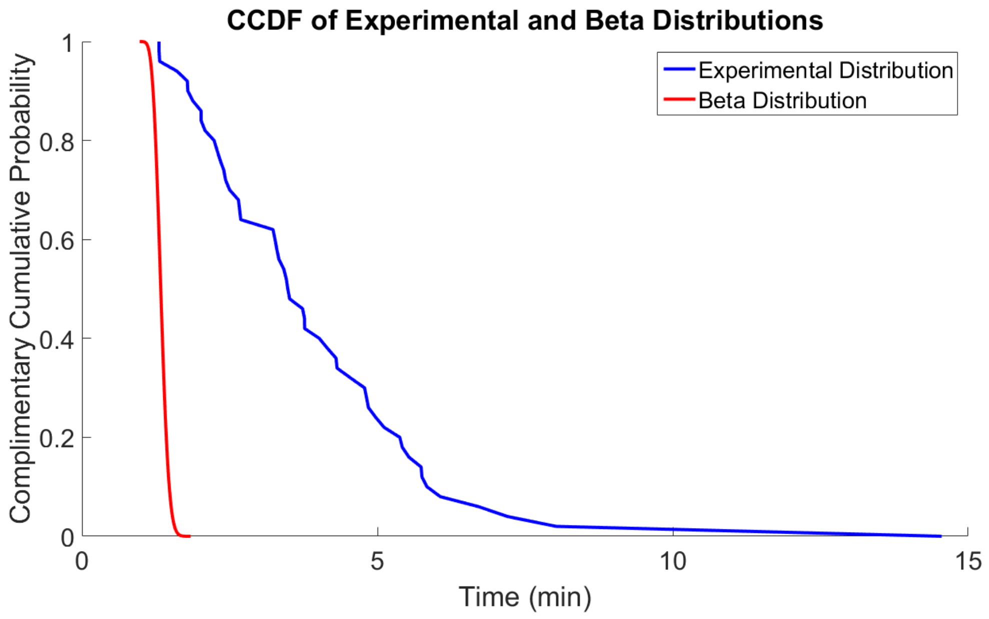

Figure 3.8: The ECCDF of experimental times to completion (blue) and the CCDF as estimated by a traditional PERT Beta approximation (red) with sample size 50.

As is evident from Figure 3.8, the two CCDF's do show much, if any, signs of coming from the same distribution. The p-value for this KS test confirms this observation with a value of 2.3467×10^{-41}, and thus the null hypothesis can be rejected at the 0.05 significance. The lack of similarity between the two models was not surprising since it has already been shown in the computational method section that any form of misinterpretation in an instruction set significantly increases the heaviness of a model tail compared with that of the standard PERT beta. However, the next step was to compare the ECCDF against the predicted model CCDF to determine whether those two models came from the same distribution. The result of which is shown in Figure 3.9.

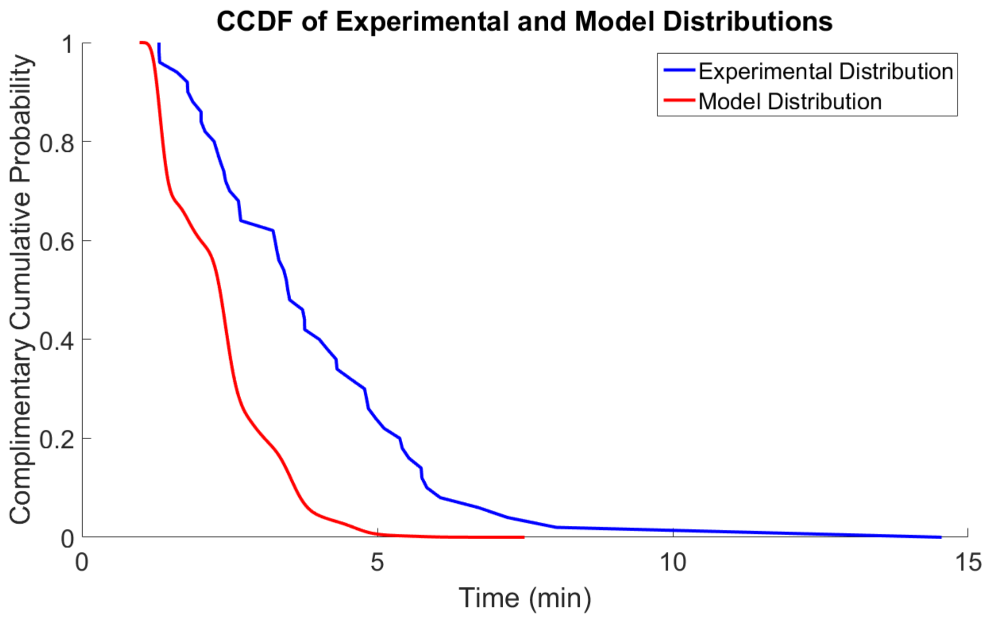

Figure 3.9: The ECCDF of experimental times to completion (blue) and the CCDF as estimated by the computational model which factored in misinterpretation (red) with sample size 50.

Fig. 3.9 shows that while, the ECCDF for the experimental times and model CCDF have closer distributions than the relationship in Fig. 3.8, the two models still do not match very closely. In fact, the experimental data results in a tail which is almost twice as long as that of the model CCDF, and with a p-value of 6.3495×10^{-10}, the null hypothesis is once again rejected at the 0.05 significance. Therefore, the initial results of the experimental tests were inconclusive in validating the hypothesis that misinterpretation in instruction sets lead to heavier tailed phenomena in project distribution estimates.

The vastly different results between the two models are most likely the result of a few different factors in the pilot study data. First, as was discussed in the previous section, the small sample size for the pilot studies resulted in time bound histograms which did not resemble a PERT beta distribution for every instruction. Additionally, the small sample size meant that very few data points for the time bounds were collected, which may not be indicative of the actual population time bounds. Since the computational model creates a distribution for each instruction based off the upper and lower time bounds for each instruction, it is imperative that those values reflect similar values for the time bounds of the population. In order to obtain values which were believed to better reflect the population bounds, the computational model was retrained, and the experimental comparison run again. The results of which are discussed in the following section.

Retraining the Model and Results

It should be evident that as the sample size for the pilot study increases, the lower time bounds will continue to decrease while the upper time bounds continue to increase in value. These bounds to continue to move until a practical limit is reached for the amount of time it takes to fold a paper airplane. Therefore, it was determined to retrain the computational model with data from the initial pilot study in addition to part of the data from the experimental studies. The decision was made to start with data from the first 10 participants from the experimental study to add to the pilot study data for a new sample size of 20 for the time bound data. Therefore, while the probabilities for each instruction remained the same, the time bounds for each instruction changed as shown in Table 3.3

The effects of adding the data for 10 experimental participants can be seen in the

Table 3.3: Estimated Time Bounds (sec) from Retrained Model

	Instruction 1		Instruction 2		Instruction 3		Instruction 4		Instruction 5	
	UB_1	LB_1	UB_2	LB_2	UB_3	LB_3	UB_4	LB_4	UB_5	LB_5
Iteration 1	41.00	7.34	55	8.05	83	9.65	49.44	4.8	61.00	6.00
Iteration 2	60.00	6.00	139.00	8.87	64.00	10.06	37.14	1.61	45.00	8.00

table as all of the time bounds were affected, with the exception of the lower time bounds for instructions 3 and 4 for multiple iterations. As was expected, the lower time bounds decreased in value while the upper time bounds increased. To visualize the effect of the new time bound data, histograms were created as before, and are shown in Figures 3.10 and 3.11

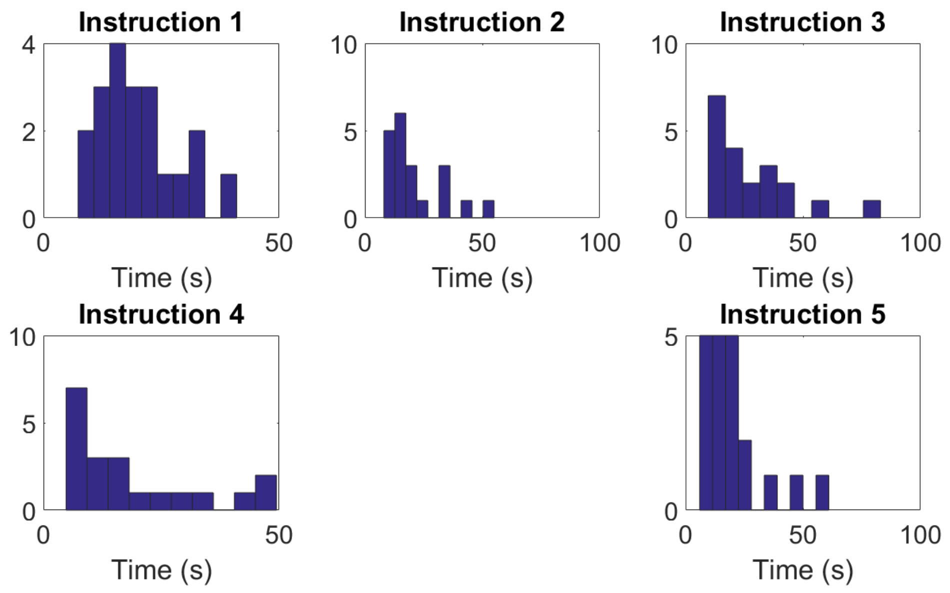

Figure 3.10: Histograms for the time distribution for the 1st iteration of the 5 different instructions with sample size 20.

The two figures show there are still a few histograms either with large outliers such as

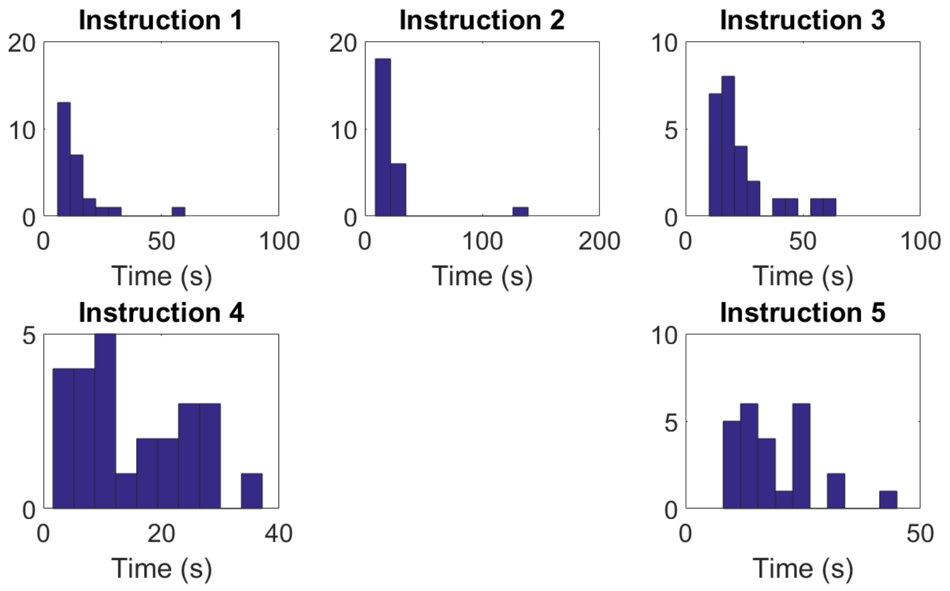

Figure 3.11: Histograms for the time distribution for any additional iterations of the 5 different instructions with sample size 20.

Instructions 1 and 2 in Fig. 3.11, or have a shape which still does not resemble a PERT beta in the strict sense such as Instructions 4 and 5 in Fig. 11. However, the addition of the 10 extra data points appears to have smoothed out the histograms for most of the instructions into shapes which resemble PERT beta distributions better than the previous histograms.

Since the time bounds changed for the computational model, a new predicted time to completion CCDF had to be created. This also meant that the sample size for the experimental data would decrease to 40 since 10 of the experimental participants were included in the pilot study data. The result of the new computational CCDF is shown in Figure 3.12

The shape of the computational CCDF as well as the estimated entropy of the instruction set is the same as before, which is to be expected since the probabilities were left unchanged in the retrained model. However, both the standard PERT beta CCDF and the

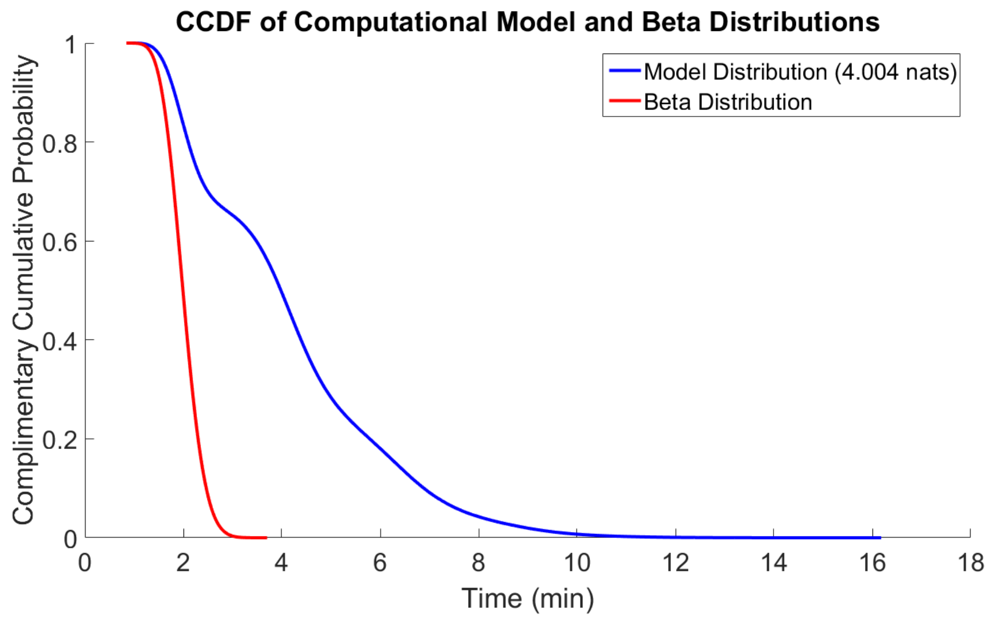

Figure 3.12: The CCDF of estimated time to completion for experimental instruction set (blue) and the CCDF as estimated by a traditional PERT Beta approximation (red) with sample size 40.

model CCDF have tails which are almost twice as long as before. This is a direct result of the significantly increased upper time bounds for each of the instructions. The two CCDF's begin at approximately the same location however, because the lower time bounds were not affected as much by the increased pilot study sample size as the upper time bounds were. As done before, the next step was to plot the new ECCDF from the experiment for the remaining 40 participants against both the standard PERT beta and computational model CCDF's, and to compare them using a KS test. Figure 3.13 shows the result of the standard PERT beta CCDF compared with the experimental ECCDF.

Although the standard PERT beta CCDF tail is twice as long with the retrained model, the two distributions in Fig. 3.13 still do not appear to be very similar. The p-value for

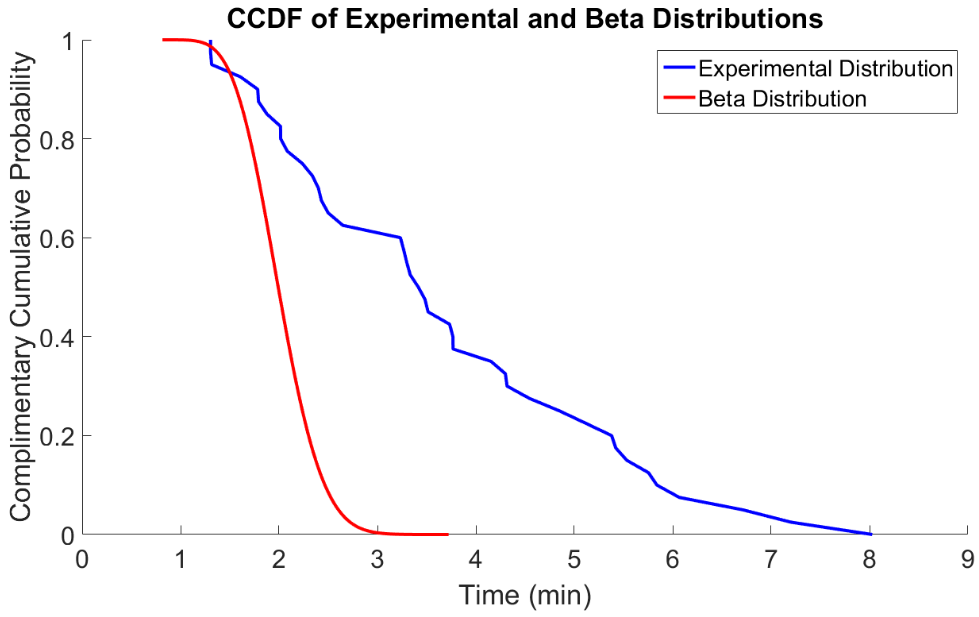

Figure 3.13: The ECCDF of experimental times to completion (blue) and the CCDF as estimated by a traditional PERT Beta approximation (red) with sample size 40.

the KS test between these two models confirms the qualitative analysis with a value of 1.4213×10^{-14}. This p-value is significantly larger than before, but is still far below α, and therefore the null hypothesis is once again rejected at the 0.05 significance. Next, the retrained model was compared with the experimental ECCDF to determine whether the two models were from the same distribution, and the result is shown in Figure 3.14

Unlike every other comparison before this, the distributions in Fig. 3.14 appear to be very similar, and in fact the KS test confirms this. The KS test for the experimental ECCDF and model CCDF results in a p-value of 0.1627, and as such the null hypothesis is plausible at the 0.05. This means that the two models could be from the same distribution, and as such the research hypothesis that misinterpretation could lead to heavier tailed phenomena in the project duration estimate can be validated with a 5% chance of making an incorrect

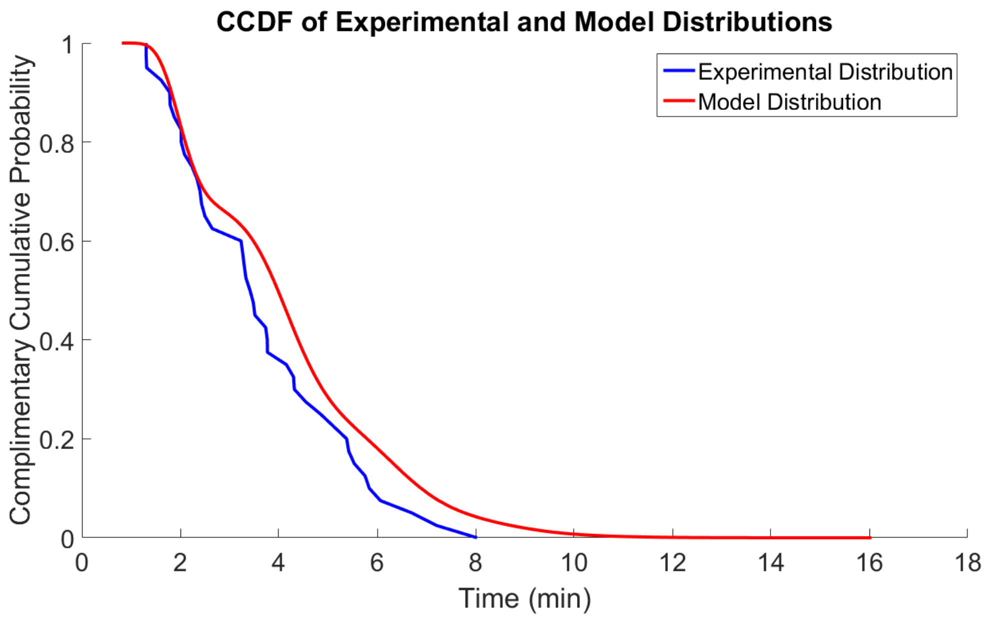

Figure 3.14: The ECCDF of experimental times to completion (blue) and the CCDF as estimated by the computational model which factored in misinterpretation (red) with sample size 40.

conclusion.

While the KS test concluded it was plausible the two models came from the same distribution, it is evident that the computational model resulted in a much heavier tail than the experimental results. Additionally, the experimental ECCDF has a similar shape to the model CCDF, however the two models parallel each other rather than intersect from about 3 minutes to 8 minutes. These observations could be the result of a few different possibilities. The first being the model assumption that the learning process was memoryless. While this assumption was understood to be a significant oversimplification of how humans learn, it was necessary in order to create the first edition of the computational model. A more accurate model would reflect the process of human learning more accurately, and as such would have more conditional probabilities which were dependent on interpretation choices from prior instructions. Another possibility is that the sample size for the experimental data was too small. While 40 participants can be considered significant in human factors studies, a larger sample size of 100-200 participants would still be more desired in order to obtain results more similar to the population. A third possible reason for the difference in the two models is due to an underestimation of the probabilities for each instruction. Although the retrained model contained data from 20 participants for the time bounds, the probabilities were left unaffected from the original 10 participants. This could lead to the probabilities being estimated more pessimistically than they actually are due to the small sample size. Another possible topic of future work would be to increase the sample size for the pilot studies in order to determine if there are more accurate values for the probabilities of each instruction than what were predicted in this experiment.

CHAPTER IV
SUMMARY AND CONCLUSIONS

Summary

This research focused on misinterpretation of instructions, and the potential impact misinterpretation had on project duration. The hypothesis for the research was that misinterpretation can go undetected for a period of time until errors it has caused are detected, and this misinterpretation leads to costly iterations in the development process which can lead to heavier tailed phenomena in the project duration distribution estimates. To validate this hypothesis, three research objectives were defined. First, to develop a computational framework to provide numerical evidence of the validity of the hypothesis. Second, to demonstrate that misinterpretation can lead to a heavier tailed distribution for a project duration. Third, to design an experiment to validate the results of the computational framework, and consequently demonstrate that the heaviness of the tail of a project duration distribution estimate can be reduced by improving the information content of task instructions.

The approach for creating the computational was to simulate the interpretation of set of instructions for completing tasks along a critical path of the design process. This was the first assumption in the model, and allowed for any parallel processes to be ignored in the creation of the model. The first research objective required making multiple other assumptions necessary to frame the specific experiment designed, and generate a computational model. First, the infinite possibilities for interpreting the instruction set for this experiment were broken down into three different categories:

1) Correct.

2) Incorrect, but plausible.

3) Grossly incorrect such that the next instruction does not make sense.

These assumptions allowed for the infinite possible end states of the experiment to be grouped into 63 distinct end states. Another key assumption was that the instruction set was memoryless, or that each interpretation for a given instruction did not depend on the previous instructions' interpretation. While this is unrealistic of how humans interpret instructions, this assumption was necessary to generate the computational model. This assumption was held in any given iteration, as well as between multiple iterations a participant was modeled to take. It was also assumed that the time to complete each instruction was an independent randomly distributed beta function based off the PERT beta method of project schedule estimation. One final assumption in the model was that the first case of misinterpretation in any iteration would be identified, and thus, that instruction's probability would change but the other instructions would retain their original probabilities. While the assumptions of the computational model severely simplify the problem, the general implications of this research are claimed to remain unaffected by the assumptions. The information content of the instruction set was calculated using information entropy as defined in information theory. Once the problem was sufficiently defined, the computational model was created as explained in Algorithms 1 and 2.

Numerical experiments were run using the computational model factoring in rework caused by misinterpretation to achieve the second research objective. The numerical experiments were designed to analyze the impact of an increasing chance of misinterpretation in an instruction on the heaviness of the distribution tail for that instruction set. Each experiment simulated an instruction set consisting of 5 instructions with 3 possible interpretations of each, corresponding with the assumption defined above. All of the experiments were conducted with a sample size of 1,000,000, and the CCDF of the computational model was compared with that of the standard PERT beta distribution to determine the effect of misinterpretation. Three experiments were run with an increasing chance of misinterpretation of 0.1%, 1%, and 10% in each instruction. For these experiments, the probabilities for the third category of interpretation corresponding to the gross misinterpretation were set to 0 in order to focus on the other two probabilities. The results from the first three numerical experiments showed that as the chance of misinterpretation in each instruction increased, and consequently the entropy of the instruction set, so too did the heaviness of the tail and overall estimated time to completion for the instruction set model CCDF. The fourth numerical experiment included the third category of interpretation, and five instruction sets were modeled with random instruction probabilities. Once again, it was observed that as the chance of misinterpretation increased, so too did the entropy of the instruction set as well as the overall estimated time to completion.

The computational model, and research objectives 1 and 2 showed the potential validity of the hypothesis from a numerical perspective, but actual experimental data was necessary in order to confirm the validity of the computational model and its corresponding assumptions. The experiment for this study was chosen to be a paper airplane experiment with 5 instructions for folding a proper airplane. In the computational model, it was assumed that the instruction probabilities and upper and lower time bounds for each instruction were known. Thus, the experimental portion of the study was broken into two categories: the pilot study, and actual experiment. The purpose of the pilot study was to generate data to estimate the probabilities and time bounds for each instruction from a small sample of participants. The limiting factor in this experiment was number of participants with a total of 60 participants for the study. Therefore, a group of 10 participants was chosen to participate in the pilot study as discussed in Section 2.3.1. From the pilot study data, the entropy of the experimental instruction set was estimated to be 4.004 nats. However, it was observed that the histograms from the time data did not resemble a PERT beta distribution for some of the instructions, and was noted as a potential indicator that a larger sample size would be needed. The other 50 participants conducted the actual experiment, and their data on times to completion as well as number of iterations was gathered and assembled in an empirical CCDF to compare with that of the computational model. To determine if the two distributions matched, a KS test was used with a null hypothesis that the two distributions matched and a significance level of α=0.05. The ECCDF of the experimental data with sample size 50 was first compared with the CCDF of a standard PERT beta estimation, and then with the CCDF of the computational model's estimation. Both KS tests resulted in the null hypothesis being rejected with p-values of 2.3467×10^{-41}

and 6.3495×10^{-10} for the standard PERT beta and computational model respectively.

The pilot study had a significantly smaller sample size than the experiment. Therefore, it was decided to retrain the computational model to include the time data from the first 10 participants in the experiment with the pilot study data to increase the pilot study sample size to 20. The upper and lower time bounds shifted for all but two of the instructions, and the probabilities for each instruction remained the same since that data was from the original 10 pilot study participants. The retrained model resulted in a standard PERT beta CCDF and computational model CCDF with tails almost twice as long as the original model due to the increased upper time bounds. The remaining sample size of 40 from the experimental data was then compared with the retrained standard PERT beta CCDF and computational model CCDF. The KS test for the experimental ECCDF and standard PERT beta CCDF resulted in the null hypothesis being rejected with a p-value of 1.4213×10^{-14}, however the ECCDF and computational model CCDF KS test resulted in the null hypothesis being plausible with a p-value of 0.1627. This result validated the numerical results of research objectives 1 and 2, and ultimately validated the research hypothesis. It was noted however, that the two CCDF's still did not match exactly, and multiple possible explanations for the difference were given in Section 3.2.3.

Future Work and Conclusions

Studies on misinterpretation could have significant impact in the realm of project management, and while this research provided some insight into how misinterpretation affects project duration estimations and times to completion, it also relied on many assumptions to simplify the problem. One of the biggest assumptions for the computational model was that the probabilities and time bounds for the instruction set were known. This is often unrealistic in many design processes, and therefore will be the first focus of future work based off this research. The same experiment will be run as in this research, but in reverse order. This will result in time data being gathered from the actual experiment with no knowledge on the probabilities of the new instruction set. Since the paper airplane experiment will be used again, the time bounds could be assumed similar since the tasks are essentially identical within the instruction set. With the other assumptions present in the computational model still holding, an optimizer will be created to identify the probabilities of each instruction, and thus which instructions are poorly written based off their risk of misinterpretation. This could simulate identifying possible ill-defined instructions in a real scenario based off historical data without having to waste resources on the current design process. Other opportunities for future work involve relaxing some of the computational model assumptions in order to make the model more complex, and simulate real world scenarios more accurately.

This research has focused on instruction misinterpretation for a very simple mechanical process, however it should be noted that misinterpretation is present at every level of communication. Whether it is peer-to-peer, supervisor-to-subordinate, human-to-computer, or any other form of communication, understanding the role of misinterpretation in the mechanical design process could greatly improve the accuracy of project schedule estimation. The results from this research provide insight into the role misinterpretation plays in the context of a simple paper airplane experiment, however the general trends discovered throughout the research could be extended into almost any situation. Thus, as computational frameworks for modeling misinterpretation become more complex and are better able to represent reality, so too will the ability to more accurately account for possible errors in the design process before they are actually made. This could have significant potential impacts on how projects are scheduled as well as modeled, and why this topic should be one researched in more depth in future studies.

REFERENCES

[1] M. Hajdu and O. Bokor, “Sensitivity analysis in pert networks: Does activity duration distribution matter?,” *Automation in Construction*, vol. 65, no. 1, pp. 1–8, 2016.

[2] P. Eremenko, “Philosophical underpinnings of adaptive vehicle make.” DARPA-BAA-12-15, Appendix 1, December 2011.

[3] K. B. I. B. M. Arena, O. Younossi and C. Grammich, “Why has the cost of fixed-wing aircraft risen?.” Report No. MG-696, Rand Corporation, 2008.

[4] D. G. Malcolm, J. H. Roseboom, C. E. Clark, and W. Fazar, “Application of a technique for research and development program evaluation,” *Operations Research*, vol. 7, no. 5, pp. 646–669, 1959.

[5] K. G. Cooper, “The rework cycle: why projects are mismanaged,” *PM network*, vol. 7, no. 2, pp. 5–7, 1993.

[6] D. N. Ford and J. D. Sterman, “Dynamic modeling of product development processes,” *System Dynamics Review*, vol. 14, no. 1, pp. 31–68, 1998.

[7] D. N. Ford, J. M. Lyneis, and T. Taylor, “Project controls to minimize cost and schedule overruns: A model, research agenda, and initial results,” in *2007 International System Dynamics Conference*, pp. 23–27, 2007.

[8] M. Giffin, O. de Weck, G. Bounova, R. Keller, C. Eckert, and P. J. Clarkson, “Change propagation analysis in complex technical systems,” *Journal of Mechanical Design*, vol. 131, no. 8, p. 081001, 2009.

[9] D. Allaire, Q. He, J. Deyst, and K. Willcox, “An information-theoretic metric of system complexity with application to engineering system design,” *Journal of Mechanical Design*, vol. 134, no. 10, p. 100906, 2012.

[10] P. J. Clarkson, C. Simons, and C. Eckert, “Predicting change propagation in complex design,” *Journal of Mechanical Design*, vol. 126, no. 5, pp. 788–797, 2004.

[11] S. Ghosh, E. Devendorf, and K. Lewis, “Exploring the effectiveness of parallel systems in distributed design processes subjected to stochastic disruptions,” *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 28, no. 04, pp. 399–412, 2014.

[12] J. D. Hey, “Do rational people make mistakes,” in *Game Theory, Experience, Rationality*, pp. 55–66, Springer, 1998.

[13] L. P. Chao and K. Ishii, “Design process error proofing: failure modes and effects analysis of the design process,” *Journal of Mechanical Design*, vol. 129, no. 5, pp. 491–501, 2007.

[14] L. Allen, R. Angel, J. D. Mangus, G. A. Rodney, R. R. Shannon, and C. P. Spoelhof, “The Hubble Space Telescope optical systems failure report,” Tech. Rep. NASA-TM-103443, National Aeronautics and Space Administration, 1990.

[15] C. Bolkcom, “V-22 Osprey tilt-rotor aircraft: Congressional Research Service Report for Congress,” tech. rep., Congressional Research Service, Jan. 7 2005.

[16] M. Hiltzik, “787 Dreamliner teaches Boeing costly lesson on outsourcing,” *Los Angeles Times*, Feb. 15 2011.

[17] C. E. Clark, “Letter to the editor-the pert model for the distribution of an activity time,” *Operations Research*, vol. 10, no. 3, pp. 405–406, 1962.

[18] C. B. G. García, J. G. Pérez, and S. C. Rambaud, “Proposal of a new distribution in pert methodology,” *Annals of Operations Research*, vol. 181, no. 1, pp. 515–538, 2010.

[19] N. R. Farnum and L. W. Stanton, “Some results concerning the estimation of beta distribution parameters in pert,” *Journal of the Operational Research Society*, pp. 287–290, 1987.

[20] B. Flyvbjerg and A. Budzier, “Why your IT project may be riskier than you think,” *Harvard Business Review*, vol. 89, no. 9, pp. 601–603, 2011.

[21] P. Mackie and J. Preston, “Twenty-one sources of error and bias in transport project appraisal,” *Transport policy*, vol. 5, no. 1, pp. 1–7, 1998.

[22] R. Atkinson, L. Crawford, and S. Ward, “Fundamental uncertainties in projects and the scope of project management,” *International Journal of Project Management*, vol. 24, no. 8, pp. 687–698, 2006.

[23] O. Morgenshtern, T. Raz, and D. Dvir, “Factors affecting duration and effort estimation errors in software development projects,” *Information and Software Technology*, vol. 49, no. 8, pp. 827–837, 2007.

[24] H. M. Gharaibeh, “Cost control in mega projects using the delphi method,” *Journal of Management in Engineering*, vol. 30, no. 5, p. 04014024, 2013.

[25] J. Hill, L. Thomas, and D. Allen, “Experts’ estimates of task durations in software development projects,” *International Journal of Project Management*, vol. 18, no. 1, pp. 13–21, 2000.

[26] M. D. Steele and W. A. Huber, “Exploring data to detect project problems,” *AACE International Transactions*, p. PM211, 2004.

[27] N.-H. Shih, “Estimating completion-time distribution in stochastic activity networks,” *Journal of the Operational Research Society*, pp. 744–749, 2005.

[28] E. D. Hahn, “Mixture densities for project management activity times: A robust approach to pert,” *European Journal of Operational Research*, vol. 188, no. 2, pp. 450–459, 2008.

[29] M. L. Martín, C. G. García, J. G. Pérez, and M. S. Granero, “An alternative for robust estimation in project management,” *European Journal of Operational Research*, vol. 220, no. 2, pp. 443–451, 2012.

[30] M. Braverman, “Suprema of compound poisson processes with light tails,” *Stochastic Processes and Their Applications*, vol. 90, no. 1, pp. 145–156, 2000.

[31] S. Asmussen and K. Binswanger, “Simulation of ruin probabilities for subexponential claims,” *Astin Bulletin*, vol. 27, no. 02, pp. 297–318, 1997.

[32] R. Grübel and R. Hermesmeier, “Computation of compound distributions i: Aliasing errors and exponential tilting,” *Astin Bulletin*, vol. 29, no. 02, pp. 197–214, 1999.

[33] A. Richards, “On upper bounds for the tail distribution of geometric sums of subexponential random variables,” *Queueing Systems*, vol. 62, no. 3, pp. 229–242, 2009.

[34] R. Keyes, *The quote verifier: Who said what, where, and when*. page 233. St. Martin’s Press, 175 Fifth Avenue, New York, N.Y. 10010, 2006.

[35] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley & Sons, 2012.

[36] T. W. Simpson, D. Rosen, J. K. Allen, and F. Mistree, “Metrics for assessing design freedom and information certainty in the early stages of design,” *Journal of Mechanical Design*, vol. 120, no. 4, pp. 628–635, 1998.

[37] J. D. Summers and J. J. Shah, “Mechanical engineering design complexity metrics: size, coupling, and solvability,” *Journal of Mechanical Design*, vol. 132, no. 2, p. 021004, 2010.

[38] H. Liu, W. Chen, and A. Sudjianto, “Relative entropy based method for probabilistic sensitivity analysis in engineering design,” *Journal of Mechanical Design*, vol. 128, no. 2, pp. 326–336, 2006.

[39] P. Krus and J. Andersson, “An information theoretical perspective on design optimization,” in *ASME 2004 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pp. 801–809, American Society of Mechanical Engineers, 2004.

[40] M. V. Koutras and S. Eryilmaz, “Compound geometric distribution of order k,” no. 1, pp. 1–17, 2016.

[41] J. N. Heineke and L. C. Meile, *Games and Exercises for Operations Management*. Prentice Hall, 1995.

APPENDIX A

MISCELLANEOUS

A.1 Experimental Instruction Set

1. With printed side face down, fold paper in half lengthwise, then open.
2. Fold top two corners towards center fold. No print should be visible.
3. Fold the angled sides towards the center fold once more.
4. Fold the paper in half along the center fold, printed side out.
5. Fold the straight portion at the top of each wing to the center fold.

APPENDIX B

EXPERIMENTAL FOLDING PAPER

Dwight Look College of ENGINEERING TEXAS A&M UNIVERSITY