Reduction of Rework at a Large Aerospace Manufacturer

by

Jeremy A. Lieberman

B.S. Material Science and Engineering, Cornell University, 2006

Submitted to the MIT Sloan School of Management and the Department of Material Science and Engineering in Partial Fulfillment of the Requirements for the Degrees of

Master of Business Administration
and
Master of Science in Material Science and Engineering

In conjunction with the Leaders for Global Operations Program at the
Massachusetts Institute of Technology

June 2012

Signature of Author ______________________________
Material Science and Engineering, MIT Sloan School of Management
May 11, 2012

Certified by ______________________________
Thomas W. Eagar, Thesis Supervisor
Professor of Materials Engineering and Engineering Systems

Certified by ______________________________
Roy Welsch, Thesis Supervisor
Professor of Statistics and Management Science, MIT Sloan School of Management

Accepted by ______________________________
Gerbrand Ceder, Professor, Materials Science and Engineering
Chair, DMSE Committee on Graduate Students

Accepted by ______________________________
Maura Herson, Director, MBA Program
MIT Sloan School of Management

This page intentionally left blank.

Reduction of Rework at a Large Aerospace Manufacturer

by

Jeremy A. Lieberman

Submitted to the MIT Sloan School of Management and the Department of Material Science and Engineering on May 11, 2012 in Partial Fulfillment of the Requirements for the Degrees of Master of Business Administration and Master of Science in Material Science and Engineering

Abstract

It is an axiom of the manufacturing of any complex product that errors will occur that require repair or discard of said product. In building aircraft, Raptor Aerospace encounters and repairs numerous deviations from the original design drawings. This process is known as rework. Reducing the amount of rework necessary represents a substantial opportunity both for improving quality and for cutting cost. Rework can be further split into several categories, with the simplest repairs referred to as reworkable discrepancies which has been valued at over $50,000,000 per year. This thesis will present a project that began at the start of the internship, when the author was paired with a specialist from Raptor Aerospace to lead a team whose purpose was to develop an approach and implement improvements that would generate a significant reduction in rework. This process would include both physical changes to the manufacturing process and would target specific aspects of the prevailing culture at Raptor.

With no existing plan for reducing rework, the two team leaders began the project by conducting a thorough analysis of existing rework data, focusing on the descriptive texts that were provided by inspectors. This analysis generated a pareto of the inspectors' words, enabling the team to identify the most common causes of rework at Raptor. Based on these results, small teams were created to perform root cause and corrective action analyses on the biggest issues. In addition to the small teams, the co-leaders also searched for solutions that would have a systemic impact on the volume of rework. To this end, an automated tool was developed that would report the rework history of every single task completed in final assembly.

Within the timeframe of the internship (6.5 months), the various approaches completed by the project team produced verified annualized savings of over $2,000,000, as well as time savings of over 40 man-hours per week. In addition, other efforts that were begun but not yet completed have anticipated savings of over $10,000,000. Finally, the project has produced indications of cultural improvements within Raptor Aerospace, as individuals and departments have begun volunteering to contribute and lead improvement efforts. Overall, it appears that the approaches taken by the project team have successfully launched a change initiative which could have substantial and long-lasting value to Raptor Aerospace.

Thesis Supervisor: Thomas W. Eagar
Title: Professor of Materials Engineering and Engineering Systems

Thesis Supervisor: Roy Welsch
Title: Professor of Statistics and Management Science

This page intentionally left blank.

Acknowledgments

I would like to thank everyone at Raptor Aerospace for their assistance with this project. These individuals, who cannot be named here, were always willing to share their opinions and advice which enabled me to make the most of my time on site. In particular, my project champion who was willing to take over the responsibility during my internship and provided the support that was necessary for the project to be a success. Further, I would like to thank my project supervisor who was a true partner throughout my time on site. His help was instrumental in the success of the project and for the future of the initiative that we started.

I would also like to thank my thesis advisors, Roy Welsch and Thomas Eagar. Both provided exceptional feedback and suggestions for completing the project and navigating the culture at Raptor. Additionally, I would like to thank Jan Klein for the guidance she provided when I was finding a new project champion.

Furthermore, I would like to thank all of my peers in the LGO program for all of the support and assistance they provided during the last two years. In particular, Bryan Drake, Matthew Reveley and Christina Williams who provided me with invaluable feedback and were willing to be a sounding board during the preparation of this thesis.

Finally I would like to thank my family for all of the support and guidance they have provided throughout my entire life. Above all others, I would like to thank my mother for her time, support, and for being the best editor that I have ever met.

Note from the Author

In the preparation of this thesis, I became concerned that a reader unfamiliar with production of complex systems or of the aerospace industry in general might be given the wrong impression about either Raptor Aerospace or the people who work at this firm. For this reason, I wanted to state up front my opinion that the people (and firm) described in this paper work exceptionally hard to provide the highest quality product possible. It was my observation that in manufacturing their product, the employees of Raptor spoke and lived the motto of quality first, making decisions that would affect profitability in order to ensure the quality of their product. As one manager put it, "the people who fly our product trust their lives to these machines." This thesis will discuss elements of Raptor's culture and processes which make it harder and more expensive for its employees to live up to this goal, but do not reduce the esteem that I hold for their efforts every single day.

This page intentionally left blank.

Table of Contents

List of Figures and Tables

Disguised Information

This thesis was prepared as the result of the author's internship experience working in close cooperation with a leading aircraft manufacturing company in the United States. In order to protect sensitive information and ensure that competitors do not gain an advantage from the information gathered, the company's name will be disguised as Raptor Aerospace. Additionally, proprietary information will be protected by disguising sensitive data, masking identifiable sources, and removing the scale on a number of graphs.

Chapter 1: Introduction

1.1 Company Background

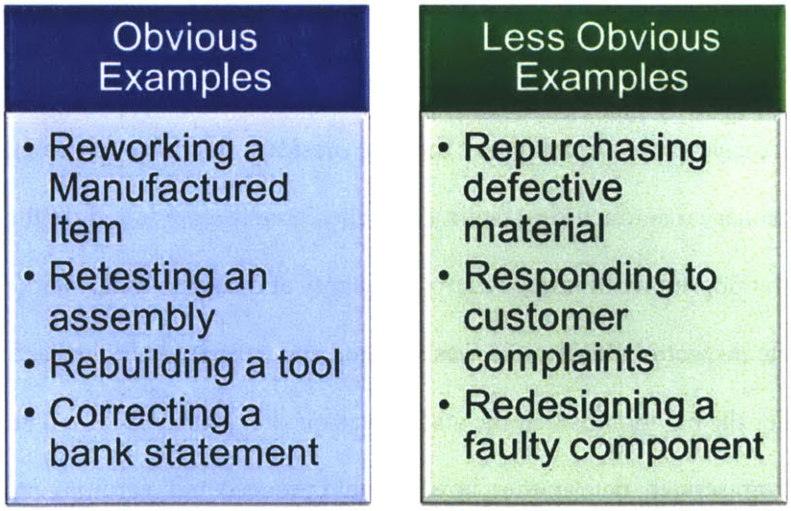
Raptor Aerospace, a large manufacturer of a variety of aircraft, was founded in the early 20th century. Since that time, the company has expanded across the globe, with facilities in over twenty locations, employing multiple thousands of people. Raptor Aerospace produces a range of products intent on caring for the individual needs of its customers. The company is divided into divisions organized around providing specific components of the aircraft, as well as final assembly, supply chain, facilities, and program offices for each of its products. Raptor originally built its entire product, except the engines, in its home factory, but over time, it has outsourced various components to other facilities that it owns, and to subcontractors. Raptor Aerospace prioritizes the safety of its product and of its working conditions, demonstrated in the company receiving awards for both throughout its history. Recently, production at Raptor has been under significant pressure to reduce the cost of every aircraft, and attention has focused specifically on the high amount of rework necessary on every aircraft manufactured.

1.2 Project Background

With an increasing focus on reducing the cost of every aircraft, Raptor Aerospace conducted an internal audit to formulate a complete breakdown of aircraft expenses. One of the largest direct costs identified by this analysis was rework, with a yearly cost of over $50,000,000. Viewing this as an opportunity to greatly reduce aircraft costs, the vice president in charge of operations and the director in charge of the primary manufacturing facility launched an initiative to reduce this expense by 90% believing it also to be an opportunity to improve aircraft quality at the same time. With these goals in mind, a specialist from the inspection department was selected and partnered with the author of this thesis to lead this project. For the 6.5 months that the author was on site, these two would be responsible for the project, launching it from scratch, determining how it would proceed and recruiting others at Raptor to assist when needed. This partnership would form the basis of a project team that would be charged with identifying opportunities to reduce rework, creating recommendations for pursuing those opportunities, and executing changes that would have a measurable and lasting impact on the amount of rework. Throughout the project, the two leads were joined for limited periods of time by others at Raptor, who provided additional assistance and expertise. However, during the time that the author was on site, the only constants on the project team were the two leaders. Going forward, for the rest of this thesis, the efforts and challenges faced by this project team will be discussed in detail providing insight into a successful approach for reducing rework at a large aerospace manufacturing firm. Once the data analysis phase was complete this project team decided to create a small team format that will be discussed in more detail in Chapter 5.

1.3 Definition of Rework and Cost of Poor Quality (COPQ)

Rework is a large and common problem in the manufacture of any complex product, such as buildings (Love, 2010), automobiles (Fisher and Ittner, 1999), ships (Clark, 2007) and aircraft (Dostaler, 2009). In the literature, several definitions for rework have been offered, all ultimately similar and addressing the same underlying concept. For convenience, we will be using the definition offered by Hegazy et al., namely that rework is the "effort of re-doing a process or activity that was incorrectly implemented the first time" (2011). In typical production, rework ranges in severity from minor, which generally can be repaired in place with normal building procedures, to severe, requiring scrapping of the immediate part as well as components that are tangentially affected by the underlying defect. Further, these more severe forms of rework often require unique engineered solutions for the individual problem, adding to the cost of repair.



Obvious Examples

- Reworking a Manufactured Item
- Retesting an assembly
- Rebuilding a tool
- Correcting a bank statement

Less Obvious Examples

- Repurchasing defective material
- Responding to customer complaints
- Redesigning a faulty component

Figure 1: Examples of COPQ (Campanella, 1999)

As Hegazy et al. pointed out, there are a number of terms, that are frequently associated with rework in the literature (2011). Some of these terms like "defects" (Harrington, 1987), "deviations" (Burati, 1992), and "deficiencies" (Axelsson, 2000), refer to the underlying conditions which lead to rework. Also closely related to "rework" is cost of poor quality (COPQ), an all-encompassing term which includes rework as defined above. COPQ is typically defined as all costs incurred because the underlying quality was not perfect (Campanella, 1999). In the Principles of Quality Costs, the American Society for Quality (ASQ) offers a number of examples of both obvious and less obvious sources of cost that stem from poor quality, as summarized in Figure 1. The relationship between these two types of costs is frequently depicted using an iceberg, where the obvious sources of cost are shown floating above the water surface, and the less obvious sources are depicted by the unseen ice beneath the surface. This type of visualization was used effectively by Krishnan, as shown in Figure 2 (2006). COPQ is generally broken into four categories; prevention costs, appraisal costs, internal failure costs, and external failure costs. Figure 30 in Appendix A includes a description of the breakdown of these costs as provided by the ASQ.

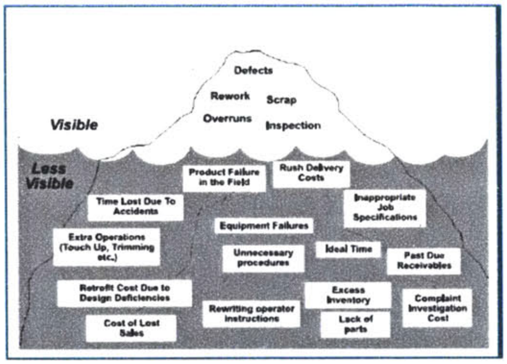

Defects
Rework
Scrap
Visible
Overruns
Inspection
Less Visible
Product Failure in the Field
Rush Delivery Costs
Time Lost Due To Accidents
Inappropriate Job Specifications
Equipment Failures
Extra Operations (Touch Up, Trimming etc.)
Unnecessary procedures
Ideal Time
Past Due Receivables
Retrofit Cost Due to Design Deficiencies
Excess Inventory
Complaint Investigation Cost
Rewriting operator instructions
Cost of Lost Sales
Lack of parts

Figure 2: COPQ Iceberg (Krishnan, 2006)

Generally, the efforts to reduce rework, COPQ or any of the other terms discussed above are lumped into improvement strategies, of which two of the most well-known are Lean and Six Sigma (Akbulut-Bailey and Motwani, 2012). Lean focuses upon the elimination of waste, while Six Sigma focused on reduction and elimination of variability by the application of statistical tools and software (Bendell, 2006). These approaches to improving manufacturing processes have been discussed in depth in the literature ever since Womack, Jones and Roos popularized Lean in *The Machine That Changed the World* (1990). Both Lean and Six Sigma are pertinent to this thesis because the methods and tools developed and implemented for this project fall under the general auspices of these approaches.

1.4 Costs of Rework

Research in several industries has found that it is very difficult to identify the true cost of rework for a number of reasons, including numerous indirect costs for rework (Love, 2002), different accounting practices for rework (Tsai, 1998), and inherent differences in the nature of rework in each industry (Omachonu, 2004). For example, this difficulty can easily be seen within the aerospace industry. Velocci Jr. found that the quoted ranges for cost of quality had a low of 2-4% of the cost of the product up to a high of 40% of the product. Going further, Velocci states, "The fact is, most companies simply do not know their cost of quality, according to some industry officials" (1998). These figures pertain to the broader cost of quality, of which rework is a significant factor. Numbers for the more specific topic of rework have not been accurately calculated at this point in time. Part of the reason why these calculations have not been completed relates to the question of direct vs. indirect costs of rework.

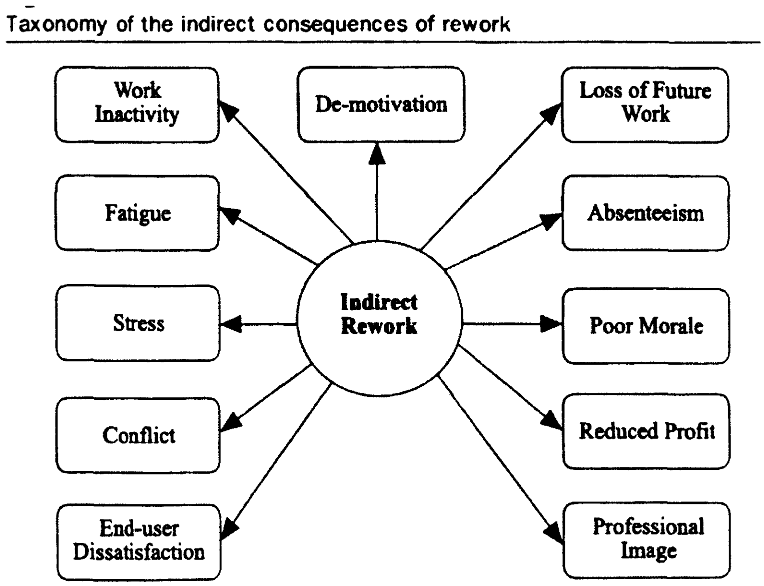


Figure 3: Sources of indirect costs of rework (Love, 2002)

Direct cost of rework typically refers to the tangible and immediately measureable costs (Love and Li, 2000; Barber et al, 2000), which includes elements like replacing scrapped parts, the man-hours needed to complete a rework task, and the cost to inspect work for defects. These costs are typically easier to measure and many cost of quality calculations are based solely on these expenses (Love, 2002). On the other hand, indirect costs are more difficult to measure. They include factors such as decreased productivity (Moselhi et al, 2005) or worker burnout (Owens et al, 2011). Love has conducted extensive work into identifying manifestations of indirect costs as demonstrated in Figure 3, but notes that "it is impossible to set aside a monetary value for each of the factors" (2002). One example of these indirect costs, noted in Figure 3, is worker inactivity, which can occur when the entire production process shuts down waiting for a replacement part. Although this example can be measured, it is often very difficult to isolate and frequently is not included in cost of quality. Overall, in his work in the building construction industry, Love found that indirect rework costs can be 3 to 6 times higher than their direct counterparts (2002).

In addition to the costs identified by Love, several other sources of indirect costs have been identified in the literature. One example that is not included in this chart is related to the costs of overtime labor. When overtime is required to make a repair, many employees are not only paid a premium, but their effectiveness is reduced, and Cooper notes that each overtime hour can have an effective cost of up to and beyond $2,000 (1994), which is equivalent to approximately $3,100 in 2012 after adjusting for inflation (Bureau of Labor Statistics, 2012). Another major indirect cost stems from the rework cycle which was introduced by Pugh-Roberts Associates and is illustrated in Figure 4 (Cooper, 1993). In the rework cycle, completed work either enters the "work really done" category or the "undiscovered rework" category where it waits to be discovered and fixed. This work can then cycle numerous times before actually being completed, creating unexpected delays and cost overruns (Cooper, 1993). The rework cycle carries a natural "quality" value, which indicates the amount of work that will enter the "undiscovered rework" category, as opposed to being completed. Cooper than continues by providing Figure 5, noting the relationship between how many times each task will pass through the rework cycle before being finished and put into the "work really done" category. This chart also shows typical values for various industries, with Aerospace tasks running through four or more cycles to complete.

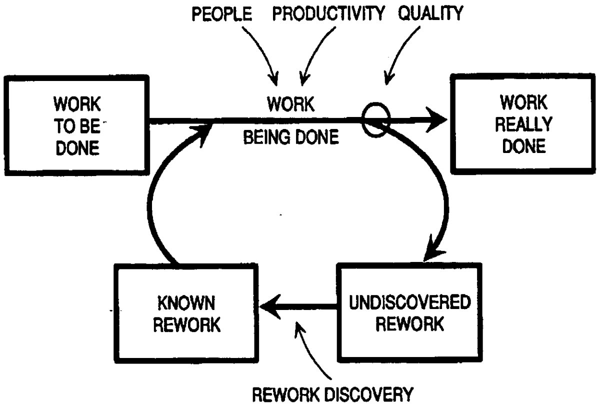


Figure 4: Rework Cycle (Cooper, 1993)

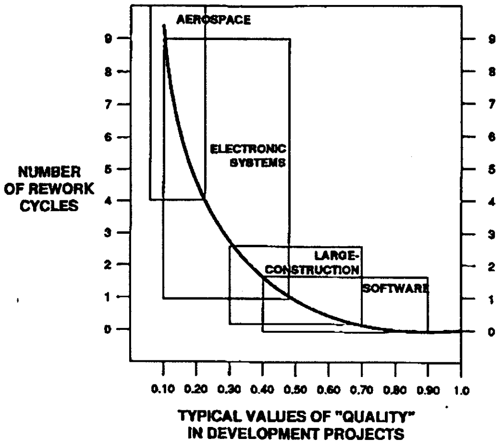


Figure 5: Number of Rework Cycles by Industry (Cooper, 1993)

While this estimate for the aerospace industry might be high, it does establish a halo effect in reducing rework as removing any source of rework will correspondingly remove multiple cycles of work.

1.5 Lean Initiatives in the Aerospace Industry

To date, extensive research has been conducted on the implementation of Lean and Six Sigma methodologies in the aerospace industry (Browning and Heath, 2009; Crute et al, 2003; Akbulut-Bailey and Motwani, 2012). These efforts have met with varying degrees of success for a variety of reasons that generally focus around cultural and political adoption of the psychology of these systems. For example, research has highlighted the importance of clear communication (Sim and Rogers, 2009), management involvement (Worley and Doolen, 2006), work rules (Drucker, 1988), the presence of a supportive culture (Achanga et al, 2006; Boyer and Sovilla, 2003) and an implementation of lean principles with an understanding of the unique requirements within each company and facility (Lewis, 2000). When Browning and Heath examined the attempt to implement lean principles on the F-22 Fighter Jet program, they observed that several tasks were evaluated to be non-value adding and were eliminated, only to be ultimately restored as necessary, as for example "tool tries" (2009). The production team for the F-22 found that these non-value adding tasks (like inspection) reduce the aircraft cost, at least in the short run, even if they do not add physical value to the aircraft.

Research on change initiatives has offered insight into the motives of individuals which can influence the success of these efforts. Boyer and Sovilla (2003) cite *Why Change Doesn't Work: why initiatives go wrong and how to try again and succeed* by Robbins & Finley (1996) in offering the following seven rules to guide an initiative:

- "People do what they perceive is in their best interest, thinking as rationally as circumstances allow them to.
- People are not inherently against change. Most will embrace initiatives, provide the change has positive meaning for them.
- People thrive under creative challenge but wilt under negative stress.
- People are different. No single elegant solution will address the entire breadth of these differences.

- People believe what they see. Actions do speak louder than words, and a history of previous deception multiplies present suspicion.
- The way to manage effective long-term change is to first visualize what you want to accomplish and then inhabit this vision until it comes true.
- Change is an act of imagination. Until the imagination is engaged, no important change can occur."

Overall, the research supports an argument that is well stated by Worley and Doolen, that "Transforming an organization to a lean enterprise is a dynamic process unique to each organization" (2006).

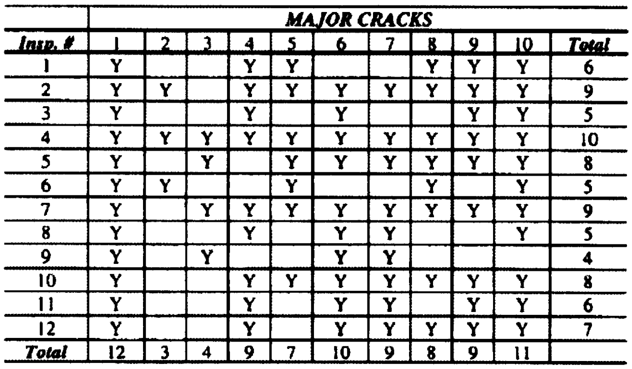
1.6 Rework and Inspection in the Aerospace Industry

In the aerospace industry, nonconforming work represents a very significant risk, even more than in many other industries, as a product failure will likely result in injury or death to the user. Such nonconforming work that leaves the factory is known as "escapes." To mitigate the risk inherent to the industry, quality is one of top priorities of leaders including Boeing (Geswein, 2011), Northrop Grumman (Grumman, 2009), UTC (United Technology Corporation, 2012), Lockheed Martin (Lockheed, 2012), Honeywell (Honeywell, 2001), and others.

One of the primary means to prevent escapes in the aerospace industry is to use multiple types of inspection to detect non-conforming work before it leaves the facility. Non-destructive testing (NDI) includes tests like eddy-current or ultrasonic technologies (Drury et al, 1997) is frequently used to assess aircraft, and remains the source of extensive and active research (De Angelis et al, 2012; Bonavolonta, 2007). However these tests cannot be used on many aspects of the aircraft; therefore visual inspection becomes necessary in over 80% of cases (Drury et al, 1997). This thesis will be focusing on visual inspection, its use within the aerospace industry, and at Raptor Aerospace specifically.

Visual inspection is an imperfect practice, as Maleyeff et al. state, "Since it is well known that inspections are not perfectly accurate, precise and unbiased, these considerations should be taken into account when designing an inspection system" (2003). The errors inherent in inspection can be divided into two categories: Type I errors, or producer's risk, which represents conforming parts which are mistakenly identified as errors, and Type II errors, or consumer's risk, which represents nonconforming parts which are mistakenly identified as in conformance. Trying to assess the size of Type II errors, Drury et al. determined that inspectors detected 68% of the opportunities placed in front of them (1997) when 12 inspectors were used to inspect components with 10 major cracks present, as illustrated in Figure 6.

In this study, inspectors were focusing on crack detection, allowing them to focus their attention on a single issue. In contrast, in most quality inspections during the final assembly, the inspector must look for hundreds of potential defect types, likely decreasing the success rate of inspectors. In Burke et al.'s work, the authors re-derived equation 1 for the true rate of nonconformance, p as follows (1995).



	MAJOR CRACKS										
Insp. #	1	2	3	4	5	6	7	8	9	10	***Total***
1	Y			Y	Y			Y	Y	Y	6
2	Y	Y		Y	Y	Y	Y	Y	Y	Y	9
3	Y			Y		Y			Y	Y	5
4	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	10
5	Y		Y		Y	Y	Y	Y	Y	Y	8
6	Y	Y			Y			Y		Y	5
7	Y		Y	Y	Y	Y	Y	Y	Y	Y	9
8	Y			Y		Y	Y			Y	5
9	Y		Y			Y	Y				4
10	Y			Y	Y	Y	Y	Y	Y	Y	8
11	Y			Y		Y	Y		Y	Y	6
12	Y			Y		Y	Y	Y	Y	Y	7
Total	12	3	4	9	7	10	9	8	9	11	

Figure 6: Inspection Results Table (Drury et al, 1997)

Equation 1:

$$p = \frac{p' - \propto}{1 - \propto - \beta} \qquad \alpha = type\ I\ errors, \qquad p = true\ nonconformance\ rate$$

$$\beta = type\ II\ errors, \qquad p' = apparent\ nonconformance\ rate$$

This is the same equation, Jaraidi et al. used to show that the probability that an item is nonconforming after *j* 100% inspections to be governed by equation 2 (1987).

Equation 2:

$$p(j) = \left[1 + \left(\frac{1-\propto}{\beta}\right)^{j} \left(\frac{1-p}{p}\right)\right]^{-1}$$

In the original application of this equation, rejected parts are discarded and do not reenter the market. In an aerospace facility, defects are fixed, either by repair or replacement of the part. Despite this difference, the equation will remain valid, because the likelihood of a rejected part being returned to the line without a repair can be assumed to be zero (before it is returned to the line, an inspector must sign off that the repair is successful). Additionally, Type I (α) errors can be assumed to be approximately zero, because any unintentionally rejected parts will not be discarded but will reenter the production steam when an installer attempts to make the repair and then finds it unnecessary. From Drury's research, the Type II (β) error rate can conservatively be assumed to be 0.38. The actual number is likely to be higher, as the greater range of criteria that inspectors are looking for will make it harder to find all defects. The last term to consider is the non-conformance rate (p) which is the hardest number to estimate. Not only will this value vary between different facilities and companies, but inspectors are generally looking for multiple criteria during a given inspection. Further, it would be inaccurate to look at the non-conformance rate for each aircraft produced, since every unit will have at least one defect, implying a non-conformance rate of 1.0 and a resulting p(*j*) of 1.0 regardless of the number of *j* inspections. Instead, Table 1 below provides p(*j*) for several values of p in order to assess the impact on the whole.

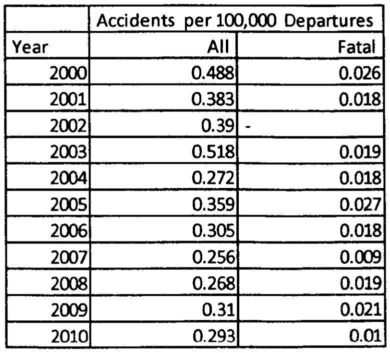
Inspection No.	p = 0.5%	p = 1%	p = 5%	p = 10%	p = 25%	p = 50%
1	0.1906%	0.3824%	1.9608%	4.0512%	11.2426%	27.5362%
2	0.0725%	0.1454%	0.7486%	1.5546%	4.3907%	11.2042%
3	0.0276%	0.0553%	0.2863%	0.5987%	1.7339%	4.7116%
4	0.0105%	0.0210%	0.1093%	0.2296%	0.6762%	1.9246%
5	0.0040%	0.0080%	0.0416%	0.0877%	0.2607%	0.7622%
6	0.0015%	0.0030%	0.0158%	0.0334%	0.0998%	0.2958%
7	0.0006%	0.0012%	0.0060%	0.0127%	0.0380%	0.1135%
8	0.0002%	0.0004%	0.0023%	0.0048%	0.0145%	0.0433%
9	0.0001%	0.0002%	0.0009%	0.0018%	0.0055%	0.0165%
10	0.0000%	0.0001%	0.0003%	0.0007%	0.0021%	0.0063%

Table 1: Inspection Type II Error Rates

While an exact number of potential components on an aircraft that could be defective is private information for each aircraft manufacturer, it is safe to assume that this number is greater than 200,000 for all major models regardless of the manufacturer. From Table 1, it is clear that the non-conformance rate has a significant impact on the number of escapes for all values of *j*. Even with a conservative nonconformance rate of 0.5%, seven inspections are needed to ensure that no more than 1 in 173,400 parts have a defect which escapes the inspection process. This success rate would still result in an average of greater than one defect per aircraft leaving the facility. One defect per aircraft turns out to be an excessively conservative estimate requiring an unreasonably high number of inspections, far above and beyond industry norms. This calculation shows that while inspection is a critical aspect of ensuring the quality of an aircraft before it is delivered, inspection will not guarantee a perfect quality aircraft, especially when considering the extremely conservative estimates used in the calculations above. By comparison, if the Type II error rate was raised by 4% up to 42%, the nonconformance rate was 1% and five inspections were performed, then approximately 1 in 7,580 parts would leave the facility with a defect (over 26 for the 200,000 components mentioned above). Therefore, it is unreasonable to believe that an aerospace manufacturer will remove all defects only through inspection.

One critical behavior that this analysis ignores which further limits the value of additional inspections is the potential for an installer risk compensation behavior in response to the increased number of inspections. Risk compensation is generally defined as the phenomenon where individuals adjust their behavior in response to perceived changes in risk, and has been researched extensively (Assum et al., 1999; Itoh et al., 2007; Phillips et al., 2011) since its introduction by Peltzman (1975), although the theory remains controversial. With regard to inspection, the theory argues that if an aerospace company increases the number of inspections, installers, knowing that an error is more likely to be caught, will be respond with less vigilance and an increased rate of errors, reducing the advantage of extra inspections.

While it may be of concern to some readers that nearly every aircraft produced will have defects that were not caught by inspection, it is important to understand the low severity of these defects. Most of the defects that are described in this section are very small and pose a



	Accidents per 100,000 Departures	
Year	All	Fatal
2000	0.488	0.026
2001	0.383	0.018
2002	0.39	-
2003	0.518	0.019
2004	0.272	0.018
2005	0.359	0.027
2006	0.305	0.018
2007	0.256	0.009
2008	0.268	0.019
2009	0.31	0.021
2010	0.293	0.01

Table 2: Aircraft Accident Rate (NTSB, 2011)

minimal risk to the performance of the aircraft, like a missing washer under a screw, and will never be noticed by either the product owner or users during the entire lifetime of the product. Additionally, aircraft manufacturers have always employed redundant systems (Osder, 1999), providing back-up protection to key systems, further protecting users. Ultimately, the performance and quality of aircraft manufacturers can be seen in the safety records of their products, as summarized in Table 2, noting the number of aircraft accidents and fatal accidents that occur every year (National Transportation Safety Board, 2011). Also, it is worth noting that these accident statistics do not exclude accidents attributed to human error like Controlled Flight into Terrain (when a pilot flies into the ground), indicating an extremely safe product.

On the whole, this analysis reaffirms Burke et al.'s statement that "the only way to ensure that only conforming product is shipped is to ensure that only conforming product is manufactured" (1995) which is the basis for the author's hypothesis.

1.7 Hypothesis

It is the author's hypothesis that a data-driven framework coupled with a cultural engagement with all levels of employees at Raptor Aerospace can lead to the development of a sustained and self-reinforcing reduction of rework, resulting in improved quality and reduced cost.

Chapter 2: Raptor Aerospace Organizational Assessment

2.1 Introduction

As discussed in Chapter 1, the large size of aerospace companies combined with the complexity of the product being manufactured has historically made it very difficult to implement change initiatives effectively. Thus, in order to test the author's hypothesis, it will be necessary to understand the current culture of Raptor Aerospace and how it will need to adjust in order to have a lasting impact on the amount of rework being completed. This chapter will explore several aspects of the current state of the company, starting with a discussion of rework and inspection. Next, the chapter will examine both the aspects of the culture which facilitate the implementation of change efforts and those which impede those same efforts. Lastly the chapter will define a desired state for a culture which will describe a Raptor positioned for continuous improvement. The targeted cultural changes will address both the long term state and the desired intermediary state that will serve as a project milestone. Chapter 4 will discuss the approaches used to work within the existing framework at Raptor while at the same time trying to adjust behaviors to match the desired future state.

2.2 Current State

2.2.1 Rework at Raptor Aerospace

Raptor Aerospace divides rework into three categories based upon the steps needed to complete the repair. The first category, dubbed within the company as reworkable discrepancies, covers small defects that can be repaired in place with standard operations. The repairs in this category are typically easier to make and include issues like missing washers or screws, chafing parts or scratches in the paint job. The next category of rework covers repairs that require special procedures that have become standardized. This work is not part of the standard construction and as such does not fit into the first category. Additionally, this type of repair is generally more expensive than that for reworkable discrepancies. The final category of rework is the most expensive and difficult repairs, requiring unique solutions to be engineered to return the product into conformance with the designs. In some circumstances, when a return to conformance is impossible, the engineering department will evaluate if there is a solution that will produce an equal quality product and if not will call for scrapping of the part. In these cases, the alternative solution s should maintain the form, fit and function of the original part.

Prior to the start of the author's internship with Raptor Aerospace, the company's management identified rework and in particular reworkable discrepancies as a significant opportunity for cost savings. Recent calculations put the direct cost of reworkable discrepancies at comfortably over $50 million, based upon the time it takes for installers and inspectors to repair and validate the acceptability of those repairs. This calculation does not include the cost of scrapping small parts, which are not tracked in sufficient detail, or of many of the indirect costs identified in Chapter 1, despite the potential for these indirect costs to be quiet large, up to six times the direct costs (Love 2002). Additionally, examples of indirect costs are readily available at Raptor as for example, one came when an installer described his practice of submitting to inspection questionable installations, knowing that the inspectors would assess the condition and determine for him if the specifications had been met (please note this is also an indication of installer risk compensation behavior described in chapter 1.5). This indirect cost of altered employee behavior adds to the cost of each individual defect, supporting the theory that Raptor might reduce overall costs further than expected by reducing the amount of rework.

Another indirect cost of rework that has not addressed in the literature, but is apparent at Raptor, is the potential for a worker conducting a fix which then creates rework in a completely unrelated, but neighboring, task. For example, some reworkable discrepancies require the repositioning of wire cables (known as harnesses). In order to complete this work, the installers will cut ties that are used to hold the wire cables together, maneuver the wires and replace the ties. Making this task more difficult is the fact that these ties are intended to hold the wires in a tight bundle and are taut. As a result, installers have reported accidentally cutting the wires in the harness while trying to remove the ties and what began as a small repair has generated a more significant and expensive follow up repair on a harness moved to allow access to a minor defect. Ultimately due to the difficulty of calculating the indirect costs, Raptor has not made an effort to assess what the potential cost ramifications might be, instead relying upon the more easily measured direct costs.

2.2.2 Inspection at Raptor Aerospace

Inspection at Raptor Aerospace demonstrates many of the limitations common to the industry described in Chapter 1. Additionally, several aspects of inspection were identified that were not widely discussed in the literature, but which impacted the performance of inspection, including variability within inspection and the role that the training program plays in that variability.

A common complaint offered by installers at Raptor about the inspection process is that it is inconsistent and different inspectors do not agree about what qualifies as a discrepancy. To support this opinion, multiple installers describe occasions when they do not agree with the rework report that is written. One common response to this circumstance is to perform no rework and resubmit the task when another inspector is around and will agree that the product conforms to the requirements, representing a case of the Type I (producers risk) error discussed in Chapter 1. In Drury and Sheehan's work, one of their inspection subjects rejected a disproportionally high number of good parts because of a poor understanding of the requirement (1968). Once Drury and Sheehan had discovered this source of extraneous rejections, they were able to remove the cause and Type I errors became negligible.

The data recording system at Raptor Aerospace makes it impossible to determine the frequency of Type I errors and thus to determine if these complaints stem from a reasonable occurrence rate or if they represent a systemic problem inflating the extent of the problem as was the case with Drury and Sheehan. Regardless of the validity of the claim, the attitude it engenders in the installers fundamentally subverts the inspection process. In some instances, installers will assume that if they cannot find the source of the rework (or disagree with it when they find it), then no problems exist and they will submit the discrepancy as repaired. When the installers are wrong, this creates delays in production and squanders inspection resources, as another inspector must reaffirm the defect.

When questioned about the issue, a group of inspectors readily acknowledged that this process occurs although they could not offer an estimate of the size of the problem. Furthermore, they identified variability in their training as a primary cause of this variation, and gray areas in the work instructions as a secondary source. Overall, these conversations indicated that there are several factors at Raptor which will increase the rate of Type I errors. However, a factor not addressed in the literature is the impact that Type I errors has on the psychology of installation crews and the potential for this mindset to increase the number of escapes when installers ignore or fail to make a repair they do not trust.

2.2.3 Training at Raptor Aerospace

When new installers and inspectors are hired, they typically receive a training program for up to two weeks to prepare them for their assignments. This training is augmented on the job by having experienced employees mentor new staff in the performance of the job. When asked about this training regimen, many of the installers and inspectors admit to remembering very little of the material covered in the course, instead relying upon the guidance given by the mentor in the performance of the job. Additionally, both installers and inspectors acknowledge that although they are supposed to have routine training courses on numerous elements of the job (e.g. tricks for difficult installs), no staff actually receives this training. In one meeting, several skilled installers and inspectors were reviewing a set of installation guidelines and discovered several tricks for a successful implementation that were unknown to everyone in the room. Love and Edwards' research into rework found that training and skill development can provide a foundation for error prevention (2004) indicating that improving the training program at Raptor would reduce rework.

From this assessment, the project team concluded that Raptor would benefit from a change to its existing training program, to one which would schedule periodic training. Such an approach combined with a system of skills qualification could benefit several aspects of Raptor's manufacturing process including rework. However, an effort to devise and implement a comprehensive training protocol would be extensive in scope and would likely encounter resistance from numerous departments within Raptor (including the training department), requiring them to cede responsibilities and budget to the new project. Furthermore, the new training program would likely need large investments of both time and capital. In spite of the value of better training, the team decided not to pursue a new protocol, believing that they were not well positioned to successful implement such an effort, and that it might undermine the entire project if it failed. Instead, where possible, the team tried to provide assistance (in time and data support) to other improvement projects which had a focus on training. Additionally, the team hoped that success in other areas of the project might build enough credibility and support to revisit the topic. In the meantime, the team would have to be aware of this training limitation when devising rework solutions.

2.2.4 Positive Cultural Elements

Employees at Raptor Aerospace demonstrate a number of positive and common attitudes which are beneficial in completing a change effort. In fact, this rework reduction project would have been impossible without relying on the benefits that stemmed from these attributes of the employees of Raptor Aerospace. Some (but not all) of these qualities are the following:

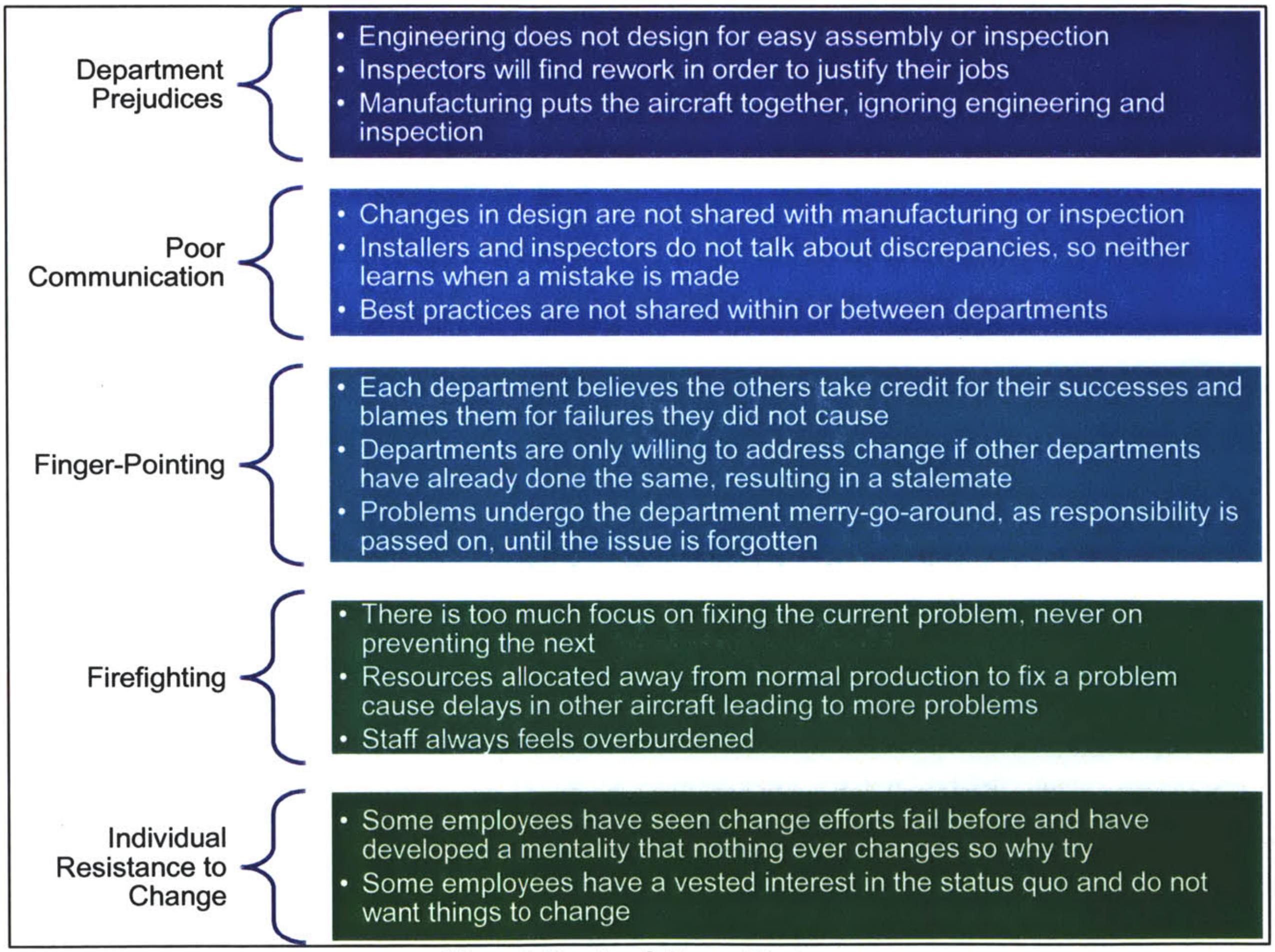
- Commitment to producing the best possible product
- A general willingness to work on improvement efforts
- Frustration at inefficiencies within their work
- Willingness to be engaged, intellectual curiosity

Going forward, this thesis will delve into negative aspects of Raptor's culture with only a limited discussion of the influences of these positive attributes. This disparity stems from the need to develop solutions that work around these attitudes but inherently draw on the positives attributes of the company's culture. None of the solutions described would be possible without many strengths inherent in Raptor

Aerospace's culture. For example, the frustration at inefficiency provided a huge motivational pillar for many employees who worked with the rework reduction team. No motivational tool was more valuable than demonstrating how rework reduction could make their jobs easier and more effective. Thus, although it will not be discussed in detail, the inherent value of these traits cannot be understated.

2.2.5 Negative Cultural Elements

In developing an assessment of the current state, the team identified a series of behaviors that inhibit natural improvements in quality, contributing to the levels of rework that were observed at the start of the project. The team broke down these behaviors into functional topics addressing a grouping of the issues that were discussed as summarized in Figure 7.



Department Prejudices

- Engineering does not design for easy assembly or inspection
- Inspectors will find rework in order to justify their jobs
- Manufacturing puts the aircraft together, ignoring engineering and inspection

Poor Communication

- Changes in design are not shared with manufacturing or inspection
- Installers and inspectors do not talk about discrepancies, so neither learns when a mistake is made
- Best practices are not shared within or between departments

Finger-Pointing

- Each department believes the others take credit for their successes and blames them for failures they did not cause
- Departments are only willing to address change if other departments have already done the same, resulting in a stalemate
- Problems undergo the department merry-go-around, as responsibility is passed on, until the issue is forgotten

Firefighting

- There is too much focus on fixing the current problem, never on preventing the next
- Resources allocated away from normal production to fix a problem cause delays in other aircraft leading to more problems
- Staff always feels overburdened

Individual Resistance to Change

- Some employees have seen change efforts fail before and have developed a mentality that nothing ever changes so why try
- Some employees have a vested interest in the status quo and do not want things to change

Figure 7: Negative Culture at Raptor

The problems causing behaviors described in Figure 7 combine to make it more difficult to complete every aspect of a change imitative effectively. In the early stages of an initiative, employees are disincentivized from participating because of the opinion that nothing changes, while later in the project finger pointing obstructs implementation. Once recommendations have been devised, the department prejudices create doubts that the other departments will live up to their commitments. Finally, even when a project overcomes these barriers, it may not be implemented by enough employees to be successful due to poor communication within the firm.

Departmental prejudices were typically grounded in an element of truth, but were frequently taken to unrealistic extremes. For example, several departments believe that manufacturing ignores the design drawings for the aircraft. When the author worked with several installers on the project, it became clear that they only looked at the drawings for tasks that were new to them or that they had not worked on recently. On the other hand, once they had performed a particular task several times, they no longer looked at those instructions and therefore frequently missed changes to the drawings that had been made by engineering. Thus, this particular prejudice appeared factual to engineering, who did not realize that the true cause was an underlying communication issue and wrongly discouraged them from making additional improvements.

A good example of finger-pointing between departments occurred well into the project during an engineering presentation that was organized by the rework reduction team. In this case, repetitive discrepancies (as will be discussed in chapter 7) had been identified as an issue of interest. Based upon this issue, the team worked with manufacturing engineering to establish a weekly meeting with senior management where engineers would present solutions they had developed for the repetitive issues identified in chapter 3.6.1. In the particular meeting in question, an engineer presented his findings on a key topic, stating that when they looked into the issue, it turned out that this one was on them, the design was not clear, and was causing a repeat problem. Here was a situation where it appeared that progress had been made in creating a culture of personal responsibility, where blame was secondary to solutions.

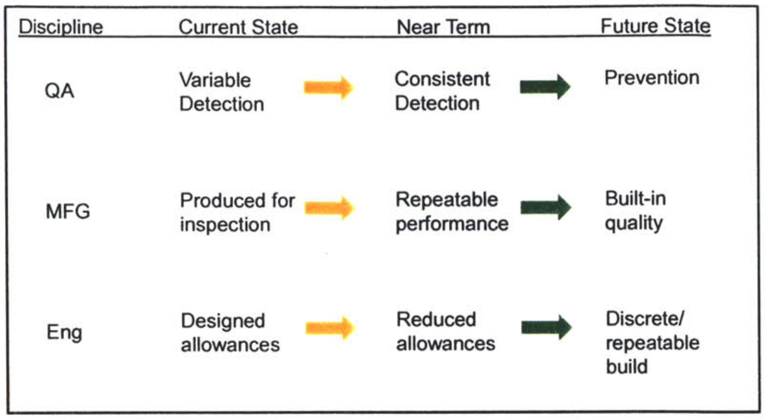
Unfortunately, before the engineer could even explain his proposed solution, one of the manufacturing managers who was present volunteered, "I'll tell you why we have an issue here; the inspectors aren't doing their job right and are just writing up discrepancies to keep themselves busy".

In short, not only did this particular manager demonstrate that he had not been paying attention to the presentation, he also angered the inspection managers who already believed that manufacturing blames inspection for every problem, regardless of the source. Fundamentally, this pattern of interaction makes change more difficult. At the same time, it also undermined the manager's credibility with executive management, because the comment ran counter to the entire conversation. This exchange demonstrates some of the underlying behavior of these departments pointing fingers at each other without trying to solve problems.

Most surprising to the author, most of the staff is aware of these cultural norms and their impact on Raptor Aerospace's improvement efforts. However, while the staff recognizes this behavior within the company, many are too focused on their own immediate fires to attempt to address the problems. Younger members of the staff seem resigned to a belief that this is how the company operates. In contrast, many of the long-term managers (who have over 20 years of experience with Raptor) describe how behavior was different when they had started with the company. In a conversation with managers from both inspection and manufacturing, each department spoke of how installers used to bring all of their discrepancies to the inspectors to learn why they occurred, and then argue their case (when appropriate). Further, it was a point of pride to the installers not to have any discrepancies and when several were written against a single installer, members of his team would encourage the installer to work harder and find out the causes for the discrepancies. Unfortunately, this behavior has since become less common and no longer represents a normal practice at Raptor.

2.3 Future State

At the beginning of the project, the team created a basic roadmap for how the different departments within operations would need to change over both the short term and long term both to create lasting changes and foster an environment of continuous improvement with respect to rework. The map is included as Figure 8 was presented to the heads of each department and to senior management for review. During the meeting it was agreed by all parties that the map represented an acceptable target and that the team would be focusing on the short term changes for the entire course of this thesis. The following sections will discuss the desired states in more detail.



Discipline	Current State		Near Term		Future State
QA	Variable Detection	→	Consistent Detection	→	Prevention
MFG	Produced for inspection	→	Repeatable performance	→	Built-in quality
Eng	Designed allowances	→	Reduced allowances	→	Discrete/ repeatable build

Figure 8: Culture Targets

2.3.1 Short Term Future State

In the short term, the focus would be on making all aspects of rework consistent and more predictable. This improvement would enable the departments to work with each other better and with less friction, by creating clear expectations for each group. This process would include an adjustment of the focus of manufacturing, inspection and engineering. Instead of just trying to complete tasks, installers would be expected to identify tasks which have inconsistencies, in order to help develop solutions. For example, if a procedure requires a wire harness to be installed but allows for several possible routes, this problem would be flagged until the single best route was selected. For inspectors, the creation of exact and consistent inspection criteria would occur so that there was no variation between inspectors as discussed in Chapter 2.2.2. In this environment, inspectors would be expected to work with each other and with the other departments to create standards to guide their own actions whenever there is variability in the standard. Finally, engineering would need to work with both inspection and installation to identify these opportunities for inconsistent installation, and with inspection in order to clarify the work instructions to remove the variability. The team believes that if the three departments can be redirected to focus on these types of priorities, it will engender a culture that encourages cooperation where continuous improvement is inherent to the workings of the company. At this point the departments would also have a framework to work together enabling them to continue down the lean path without the acrimony that has previously been a hallmark of their interactions.

2.3.2 Long Term Future State

The goal for the long term was to convert the manufacturing and inspection process from one of verification of quality into one that improves quality. This future state would have different expectations and responses for each department, both individually and working as a team. First, manufacturing would be more focused on built-in quality, identifying occasions where aspects of the assembly make this harder and always working to develop new ideas and methods to enhance quality. Secondly, engineering would continually be working to tighten tolerances and standardize the assembly to make the job easier for both inspection and manufacturing. Finally, inspection would undergo the most radical change, becoming a department with a priority of preventing rework rather than just identifying where it was needed. This would give inspectors a personal stake not just in identifying discrepancies but also in developing solutions for how to make each following aircraft better.

In this future state, every discrepancy would become an opportunity for manufacturing, inspection and engineering to team up and form a mini root-cause and corrective-action team. In this way, all three would share credit for improving the quality of the aircraft with a clear path for removal of discrepancies. Unfortunately, to get to this point, significant success to reduce rework would be needed, just to free up time and capacity that is currently being spent fixing each occurrence of rework, and none of the departments are able to look beyond the current aircraft.

Chapter 3: Data Analysis

3.1 Organization of Chapter

This chapter will focus on the data analysis that was conducted throughout the project. It will begin with a discussion of the sources and types of data that were available for analysis, followed by a detailed discussion of the limitations present in that data, which had impacted previous efforts to extract value from the dataset. Next, the chapter will focus on the different ways that the data was analyzed to generate value for the project team. Finally, there will be a discussion of using the data effectively to reduce the amount of rework at Raptor Aerospace.

3.2 Data Sources

Raptor Aerospace is a data rich company, which documents and records every reworkable discrepancy that occurs in the manufacture of an aircraft. This data is maintained by the IT department on a public server which contains databases for numerous departments within the company. Frequently, the names of fields in these databases are in shorthand, making them difficult to understand by anyone not familiar with the underlying data. The records for each discrepancy are stored in the production dataset and consist of detailed information about the specific defect which is intended to assist with data analysis, and in most instances, are entered into Raptor's electronic database. This information includes, but is not limited to, the following:

- Multi-digit defect code
- Detailed description of defect
- Location and aircraft of the defect
- The task that resulted in the defect
- Date of task completion
- Date of defect identification
- Name of the inspector who identifies the defect

- Name of the installer who created the defect

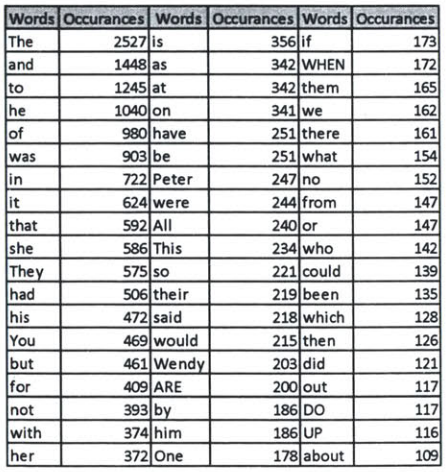
3.3 Data Limitations

There are several major limitations with the dataset at Raptor Aerospace which are relevant for this project. The first limitation in the data is the inaccuracy of the multi-digit defect code. Research into the impact of having too many options, known as "choice overload', has shown that choosers can become demotivated from making a choice when faced with too many options (Iyengar and Lepper, 2000). In the case of inspectors, faced with too many defect code options, they frequently use either the most generic code numbers or the first code on the list. Also related to the data entry, the procedures at Raptor require the inspectors to write a unique report for every defect they observe. However, inspectors frequently combine defects in close proximity that are discovered at the same time into a single record, thereby underreporting the total number of discrepancies. Further limiting the value of the dataset in most circumstances, the only name attached to the installation is that of the person providing the final approval. Frequently, an inspector provides this final step, making it impossible to determine which installer actually performed the work. Another limitation to the data is that rework in the final inspection (known as the final shake) is recorded manually and is never entered into the electronic system, preventing these files from being analyzed efficiently. Additionally, rework data from key suppliers is also unavailable, making it impossible to examine the entire value stream for common defect causes. The final limitation in the data concerns defects created by a supplier but discovered by Raptor Aerospace. The electronic system at Raptor does not allow the inspectors to record these defects against the appropriate supplier; instead they are typically written against tasks completed in the first position at Raptor.

The combined effect of the limitations noted above have historically made it difficult to analyze the discrepancy data and identify common causes that can be targeted for elimination. As a result, the first step for the project team was to analyze the data and develop a methodology for attacking the problem.

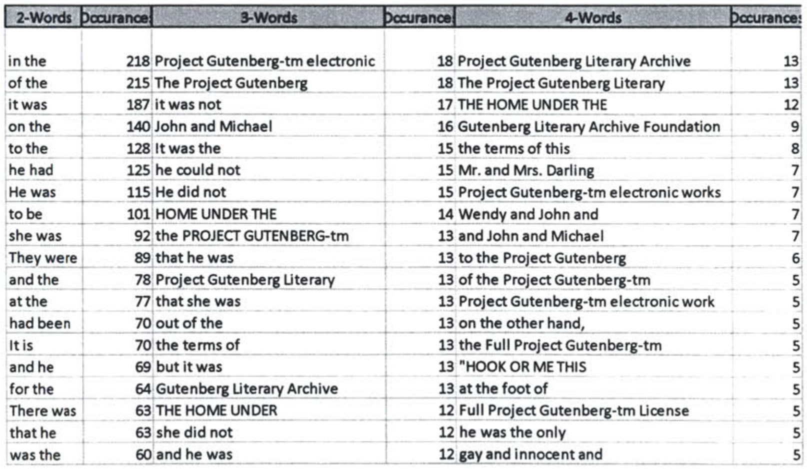
3.4 Primary Data Analysis Methodology and Results

As previously stated, the inaccuracies of the multi-digit defect code make it difficult to identify common causes of discrepancies. Therefore, the project team decided to focus on the detailed defect description provided by the inspectors. These descriptions range in length from one sentence up to a paragraph, and are intended to enable another person to identify and understand the observed deficiency, and are written in the individual author's vernacular. To determine the most frequent problems, the team developed a tool which could parse through millions of lines of text, creating a word pareto identifying the most common words in the entire recordset, along with the most common two-word, three-word and four-word combinations. This tool was programmed primarily using visual basic for applications (VBA) and built-in functionality of Microsoft Access. One of the requirements that the team imposed upon this tool (and on other tools that were created and will be discussed later) is that the tool be automated to minimize operation time and simplify use in the future. As a result, once the dataset to be analyzed has been imported into the tool, a single button press will operate the entire analysis and output the final results. A selection from the word pareto using the text from *Peter Pan* by J.M. Barrie as an input is included in Figure 9 and 10 (Project Gutenberg, 2008). A copy of the underlying code can be found in Appendix B



Words	Occurances	Words	Occurances	Words	Occurances
The	2527	is	356	if	173
and	1448	as	342	WHEN	172
to	1245	at	342	them	165
he	1040	on	341	we	162
of	980	have	251	there	161
was	903	be	251	what	154
in	722	Peter	247	no	152
it	624	were	244	from	147
that	592	All	240	or	147
she	586	This	234	who	142
They	575	so	221	could	139
had	506	their	219	been	135
his	472	said	218	which	128
You	469	would	215	then	126
but	461	Wendy	203	did	121
for	409	ARE	200	out	117
not	393	by	186	DO	117
with	374	him	186	UP	116
her	372	One	178	about	109

Figure 9: Single Word Sample Pareto



2-Words	Occurance	3-Words	Occurance	4-Words	Occurances
in the	218	Project Gutenberg-tm electronic	18	Project Gutenberg Literary Archive	13
of the	215	The Project Gutenberg	18	The Project Gutenberg Literary	13
it was	187	it was not	17	THE HOME UNDER THE	12
on the	140	John and Michael	16	Gutenberg Literary Archive Foundation	9
to the	128	It was the	15	the terms of this	8
he had	125	he could not	15	Mr. and Mrs. Darling	7
He was	115	He did not	15	Project Gutenberg-tm electronic works	7
to be	101	HOME UNDER THE	14	Wendy and John and	7
she was	92	the PROJECT GUTENBERG-tm	13	and John and Michael	7
They were	89	that he was	13	to the Project Gutenberg	6
and the	78	Project Gutenberg Literary	13	of the Project Gutenberg-tm	5
at the	77	that she was	13	Project Gutenberg-tm electronic work	5
had been	70	out of the	13	on the other hand,	5
It is	70	the terms of	13	the Full Project Gutenberg-tm	5
and he	69	but it was	13	"HOOK OR ME THIS	5
for the	64	Gutenberg Literary Archive	13	at the foot of	5
There was	63	THE HOME UNDER	12	Full Project Gutenberg-tm License	5
that he	63	she did not	12	he was the only	5
was the	60	and he was	12	gay and innocent and	5

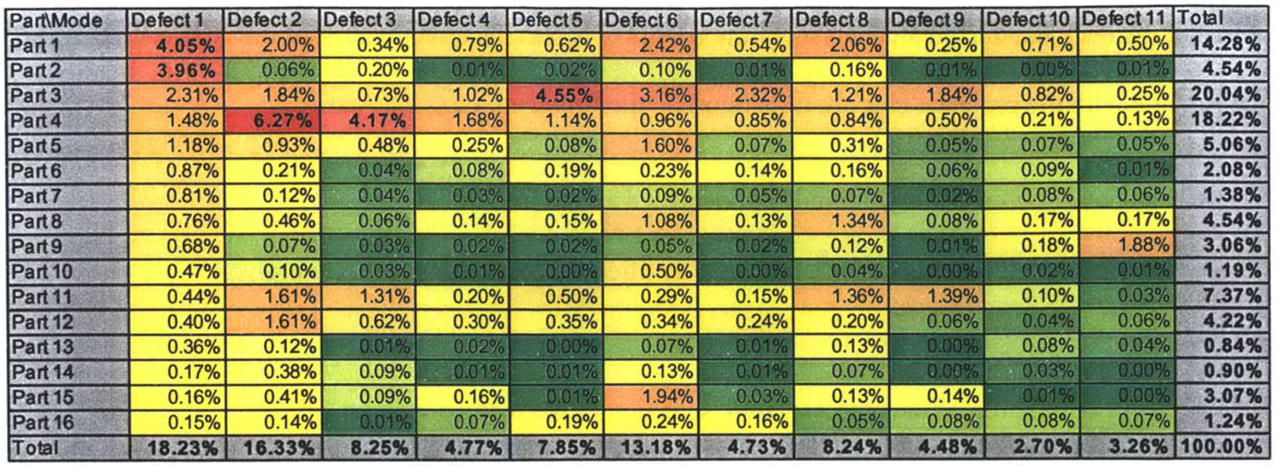
Figure 10: Multi Word Sample Pareto

along with a more detailed discussion of its operation.

As can be seen in the sample word pareto, the most frequent words in any text analysis will be prepositions and other common words (e.g. the, of, or, and, etc.). Once these terms were removed, the team was left with a list of key words that could be split into two categories, defect mode and parts. Some of these terms could then be aggregated into common groupings, such as oversized and undersized (wrongsized). Additionally, certain terms were frequently misspelled and these misspellings were combined with the correct spelling to form a complete category. Examples of some terms which occurred in each category are listed below:

- **Defect Mode:** Chafing, Loose, Missing, Preload, Wedged, and Wrongsized
- **Parts:** Clamp, Hardware, Harness, Sealant, and Washer

With these terms identified, the team created a Microsoft Access tool augmented with VBA to analyze the entire dataset in several different directions looking for potential approaches for reducing rework. A limited version of this tool is included in Appendix C. This tool also incorporates additional information from a separate file which records the time of each line roll, to be compared with the dates stored within the reworkable discrepancy dataset, confirming the location where each defect is made and identified. As part of the initial analysis, each record that contained one of the key terms identified by the word pareto was flagged. In this way, records containing multiple defects would be counted for each defect that was recorded, but would only be counted once regardless of how many times a key word appeared in the record. The tool then created a summation of the different defect modes vs. the different parts which the team viewed as potentially actionable deficiencies. The following table summarizes the relative importance of each summation:



Part\Mode	Defect 1	Defect 2	Defect 3	Defect 4	Defect 5	Defect 6	Defect 7	Defect 8	Defect 9	Defect 10	Defect 11	Total
Part 1	**4.05%**	2.00%	0.34%	0.79%	0.62%	2.42%	0.54%	2.06%	0.25%	0.71%	0.50%	**14.28%**
Part 2	**3.96%**	0.06%	0.20%	0.01%	0.02%	0.10%	0.01%	0.16%	0.01%	0.00%	0.01%	**4.54%**
Part 3	2.31%	1.84%	0.73%	1.02%	**4.55%**	3.16%	2.32%	1.21%	1.84%	0.82%	0.25%	**20.04%**
Part 4	1.48%	**6.27%**	**4.17%**	1.68%	1.14%	0.96%	0.85%	0.84%	0.50%	0.21%	0.13%	**18.22%**
Part 5	1.18%	0.93%	0.48%	0.25%	0.08%	1.60%	0.07%	0.31%	0.05%	0.07%	0.05%	**5.06%**
Part 6	0.87%	0.21%	0.04%	0.08%	0.19%	0.23%	0.14%	0.16%	0.06%	0.09%	0.01%	**2.08%**
Part 7	0.81%	0.12%	0.04%	0.03%	0.02%	0.09%	0.05%	0.07%	0.02%	0.08%	0.06%	**1.38%**
Part 8	0.76%	0.46%	0.06%	0.14%	0.15%	1.08%	0.13%	1.34%	0.08%	0.17%	0.17%	**4.54%**
Part 9	0.68%	0.07%	0.03%	0.02%	0.02%	0.05%	0.02%	0.12%	0.01%	0.18%	1.88%	**3.06%**
Part 10	0.47%	0.10%	0.03%	0.01%	0.00%	0.50%	0.00%	0.04%	0.00%	0.02%	0.01%	**1.19%**
Part 11	0.44%	1.61%	1.31%	0.20%	0.50%	0.29%	0.15%	1.36%	1.39%	0.10%	0.03%	**7.37%**
Part 12	0.40%	1.61%	0.62%	0.30%	0.35%	0.34%	0.24%	0.20%	0.06%	0.04%	0.06%	**4.22%**
Part 13	0.36%	0.12%	0.01%	0.02%	0.00%	0.07%	0.01%	0.13%	0.00%	0.08%	0.04%	**0.84%**
Part 14	0.17%	0.38%	0.09%	0.01%	0.01%	0.13%	0.01%	0.07%	0.00%	0.03%	0.00%	**0.90%**
Part 15	0.16%	0.41%	0.09%	0.16%	0.01%	1.94%	0.03%	0.13%	0.14%	0.01%	0.00%	**3.07%**
Part 16	0.15%	0.14%	0.01%	0.07%	0.19%	0.24%	0.16%	0.05%	0.08%	0.08%	0.07%	**1.24%**
Total	**18.23%**	**16.33%**	**8.25%**	**4.77%**	**7.85%**	**13.18%**	**4.73%**	**8.24%**	**4.48%**	**2.70%**	**3.26%**	**100.00%**

Figure 11: Defect vs Part Cross Pareto

With a series of potential opportunities identified, the tool also overlaid the locations of the deficiencies onto an image of the aircraft to identify potential hotspot locations. This overlay was also completed for each of the top five Part-Defect combinations, although these heat maps cannot be included in this thesis due to their sensitive nature. Next, the top five defects were plotted over time, confirming that these topics were still large and active opportunities for improvement. Samples of two of these plots are shown in the chart below. Also included is a copy of the chart for all defects to compare the overall trends.

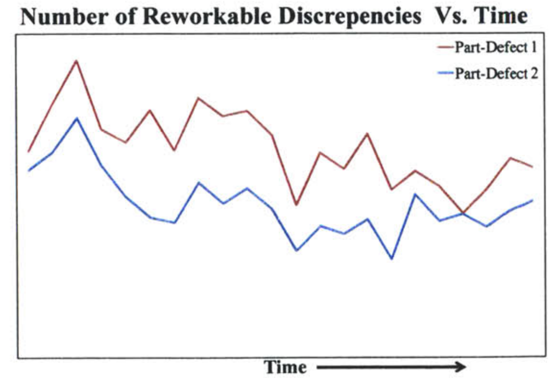


Figure 12: Sample Part vs Defect Time Chart

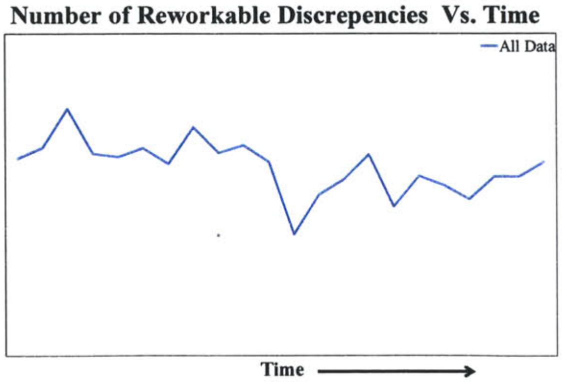


Figure 13: Discrepancies vs Time

These charts showed that the topics remained large issues although there were small downward trajectories in the data, which the team associated with learning curves and specific external influences, such as changes in the line organization or the number of aircraft being manufactured.

Finally, the team conducted an analysis of the results to assess the effectiveness of this approach before continuing with the project. This analysis returned the following facts and statistics:

- 90.56% of all records contained an identified key term
- 56.65% of all records contain both an identified defect and an identified part
- Records contain a range of 0-12 key terms
- An examination of the representative selection of the records without key terms found the following typical causes:
    - The defects or parts described were not considered high drivers
    - The description was generic and could not be attributed to specific parts or defects
    - Specific part numbers were used in the description instead of the name of the part (e.g. Part 333333 instead of "Washer")

With over 90% of the records containing one of the identified key terms, the project team and Raptor's management believe that the team's approach to data analysis was comprehensive and effective. The tools that were created will enable the company to revisit the data in the future to identify new high drives, once the current ones have been successfully eliminated. Furthermore, to test the scalability of the tools, they have processed test record sets of over 350,000 distinct files, when they performed a complete analysis with a completely automated run times of approximately 30 minutes.

3.5 Considerations for Primary Data Analysis

One fear that the project team had about the data analysis approach used in this project was the potential to count the number of each type of discrepancy inaccurately. In particular, by flagging a record for every key word present, the records will tend to over-count the number of each type of discrepancy. For example, if a particular record stated "the washer next to a particular clamp is missing," then the data analysis tool would count that record in both the missing washer category (accurate) and the missing clamp category (inaccurate). These Type II errors are balanced by Type I errors related to the individual language used by each inspector. An example of a Type I error occurred when an inspector wrote "a cl asbly is missing." A person could recognize that he is referring to a clamp assembly, but the program will omit this particular record because the letter combination "cl" may not always refer to a clamp and thus could not be included in the program. Ultimately, the team assessed that both the Type I and Type II errors were individually less than a few percent. Because they were acting in opposing directions, the team concluded that they would not have a significant impact on the size of each improvement opportunity, particularly given the number of data points in the full dataset.

Another consideration of the project team was how to manage the erroneous multi-digit defect code mentioned in the previous section. The project team discussed several possible alternatives to this code, including having inspectors select from a list of the key words identified by the program. Then this concept was ultimately spun into its own project, managed by the quality organization. Thus, their efforts to address this issue are not discussed within this thesis.

3.6 Secondary Data Analysis

After the primary data analysis phase was complete, a number of targeted project teams were established to address specific topics, the format and structure of which will be discussed in the next chapter. At the request of these teams, additional data analysis was done to identify potential other opportunities. The rest of this chapter will be devoted to a discussion of this secondary data analysis.

3.6.1 Repetitive Reworkable Discrepancies

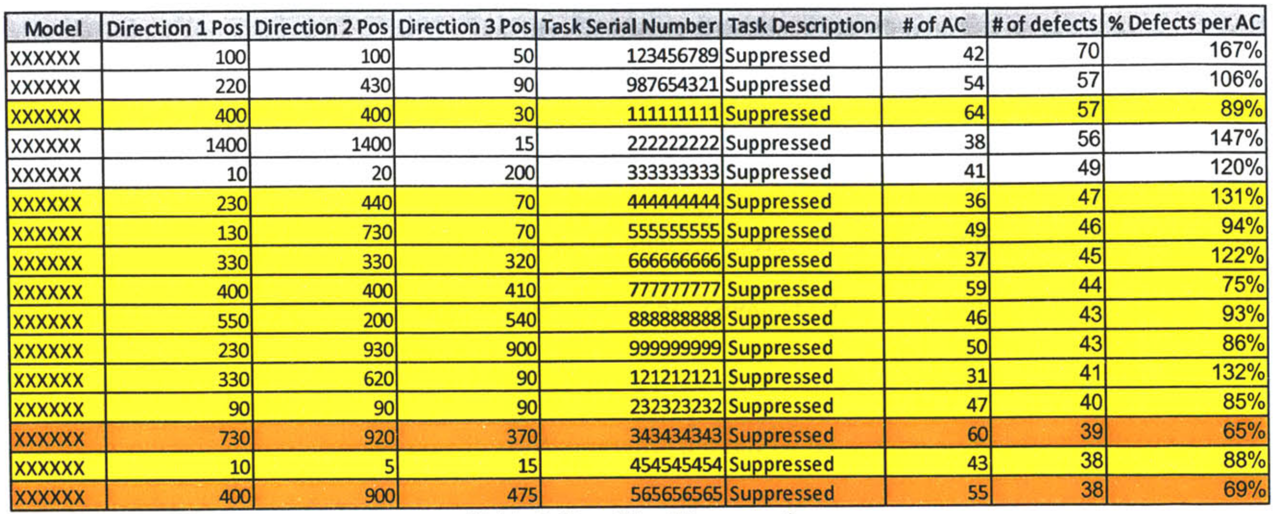
One of the concerns raised later in the project was the potential existence of repetitive issues that occur on every aircraft. The discussions that led to this concern, as well as the use of the results discussed in this section, will be addressed in greater detail in Chapter 8, which focuses on the second defect-part team. This section provides the data analysis that was conducted to address the repetitive issues.

Establishing that a particular discrepancy occurs multiple times is actually quite difficult. When different inspectors observe and record the same discrepancy on different aircraft, they will use different language, making it harder to recognize the similarities (also true when the same inspector sees an issue on different days). Additionally, many tasks require the installation of several similar parts, such as installing four clamp assemblies near each other. In this case, it is important to distinguish between the clamp assemblies, as each may have a unique cause, which would make aggregating their discrepancies misleading.

In order to identify potential examples of repetitive deficiencies, the team programmed an extension to the keyword analysis already completed. This extension sorts the data into separate buckets based upon the following criteria:

- Location: Each bucket would only contain deficiencies within a five inch cube
- Task: Only deficiencies written against the same task
- Model: Only deficiencies against the same model aircraft
- Number of Aircraft: Only deficiencies occurring on at least ten aircraft

Once sorted into these buckets, the number of aircraft affected and the number of discrepancies for each key word would be summed up. Next, the program would look for the biggest issue with each bucket and would determine the percentage of aircraft affected by this problem. In the event that multiple discrepancies were written against the same aircraft, task, location and with the same key words, both discrepancies would be counted (this can result in over 100% of affected aircraft having a repetitive defect). To finish the exercise, the data was then sorted by the maximum number of repetitive discrepancies. The following Table summarizes a selection of these results, although the individual data has been disguised.



Model	Direction 1 Pos	Direction 2 Pos	Direction 3 Pos	Task Serial Number	Task Description	# of AC	# of defects	% Defects per AC
XXXXXX	100	100	50	123456789	Suppressed	42	70	167%
XXXXXX	220	430	90	987654321	Suppressed	54	57	106%
XXXXXX	400	400	30	111111111	Suppressed	64	57	89%
XXXXXX	1400	1400	15	222222222	Suppressed	38	56	147%
XXXXXX	10	20	200	333333333	Suppressed	41	49	120%
XXXXXX	230	440	70	444444444	Suppressed	36	47	131%
XXXXXX	130	730	70	555555555	Suppressed	49	46	94%
XXXXXX	330	330	320	666666666	Suppressed	37	45	122%
XXXXXX	400	400	410	777777777	Suppressed	59	44	75%
XXXXXX	550	200	540	888888888	Suppressed	46	43	93%
XXXXXX	230	930	900	999999999	Suppressed	50	43	86%
XXXXXX	330	620	90	121212121	Suppressed	31	41	132%
XXXXXX	90	90	90	232323232	Suppressed	47	40	85%
XXXXXX	730	920	370	343434343	Suppressed	60	39	65%
XXXXXX	10	5	15	454545454	Suppressed	43	38	88%
XXXXXX	400	900	475	565656565	Suppressed	55	38	69%

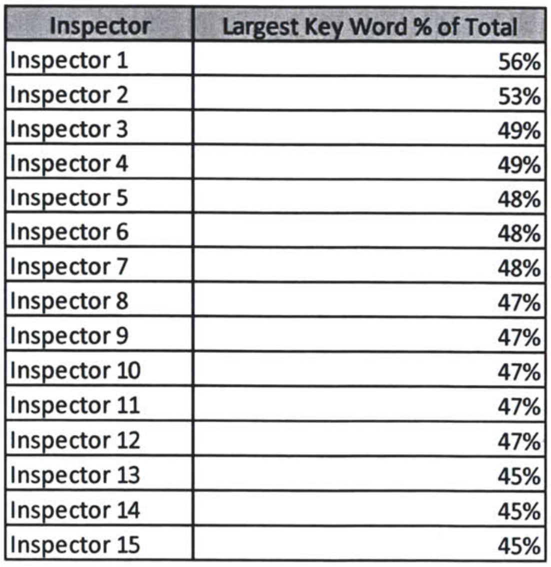
Table 3: Repetitive Discrepancies

The data in Table 3 was divided into three categories. The first category, which is not colored, identified a topic that had been selected for a target team. The second and third categories, colored yellow and orange respectively, identified a series of similar repetitive issues that were the key results of this analysis.

From this list, the team selected ten trial issues and sorted through each defect record in order to remove any that might be extraneous. The trial issues were selected based upon a consideration of both the size of the error and their similarity, in order to increase the likelihood that the same general solution method could be applied to each topic. Analysis of these issues found that less than 5% of the records were identified as not being caused by this particular repetitive deficiency and these data points were removed. At the same time, the team was unable to sort through the records that were not selected by the program (to determine if more should be included) due to the size of the full database and the large number of potential errors that might cause a record to be neglected. For more information regarding the use of this analysis, see Section 7.3.1.2.

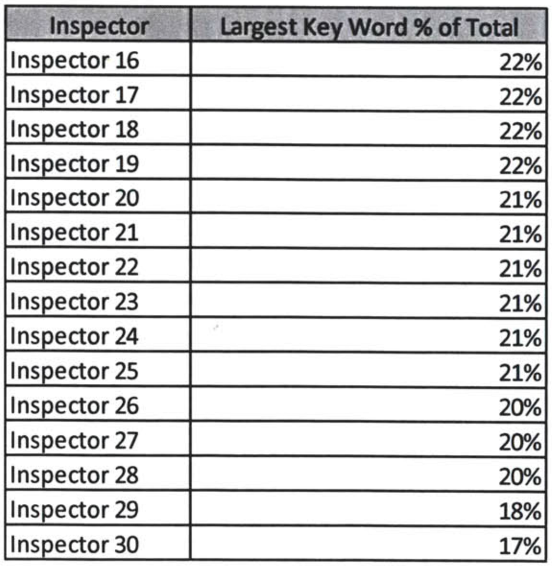
3.6.2 Specific Inspector Analysis

During a conversation with the project team, two manufacturing managers raised a concern that individual inspectors had pet defect types that they focused on, inspiring a probe of the data to determine if individual inspectors frequently exhibit this behavior. Research into inspection has found numerous examples of different types of bias (Maleyeff et al, 2003; Mei et al, 1975). However, no research could be located on the specific question of inspectors focusing on the same issues. In response, the team built an additional extension onto the keyword analysis, similar to the method used in section 3.6.1. This extension adds an additional filter and sort to the data, creating a breakdown of the total number of discrepancy reports and what key words were used in those reports for each inspector. In order to prevent this data from being used either to promote or to dissuade the writing of discrepancy reports, this intermediary state was kept concealed within the program. Next, the program determined the percentages of each category of discrepancy that was written (removing the total number from the data file). We also filtered out certain inspectors because of particular aspects of their job that would lead to an above average concentration of defects in a single topic (e.g. an inspection requiring special skills that only certain inspectors can perform may lead to a structural bias). A selection of the results for the remaining inspectors is presented in the Tables 4 and 5.



Inspector	Largest Key Word % of Total
Inspector 1	56%
Inspector 2	53%
Inspector 3	49%
Inspector 4	49%
Inspector 5	48%
Inspector 6	48%
Inspector 7	48%
Inspector 8	47%
Inspector 9	47%
Inspector 10	47%
Inspector 11	47%
Inspector 12	47%
Inspector 13	45%
Inspector 14	45%
Inspector 15	45%

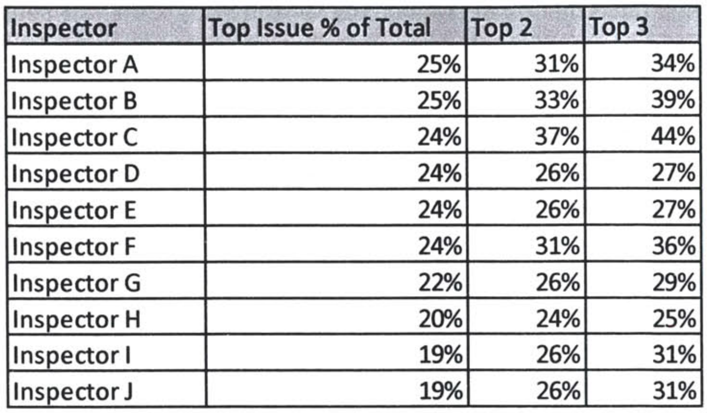
Table 5: Inspector Pet Topics



Inspector	Largest Key Word % of Total
Inspector 16	22%
Inspector 17	22%
Inspector 18	22%
Inspector 19	22%
Inspector 20	21%
Inspector 21	21%
Inspector 22	21%
Inspector 23	21%
Inspector 24	21%
Inspector 25	21%
Inspector 26	20%
Inspector 27	20%
Inspector 28	20%
Inspector 29	18%
Inspector 30	17%

Table 4: Inspector Pet Topics (cont.)

In section 3.6.1, the primary data analysis indicated that the key word responsible for the most discrepancies represented approximately 20% of the total. By contrast, well over fifteen inspectors wrote over 40% of their discrepancy reports against a single key word. At the other end of the spectrum, over fifteen inspectors had a largest key word that was in line with the overall total from the primary data analysis, suggesting that these inspectors were unbiased sources. However, when the data was examined more closely, it was apparent that these inspectors did not have the same distribution as the combined data. To expand on this possibility, the data was broken down based upon combined part-defect topic area producing the Table 6.



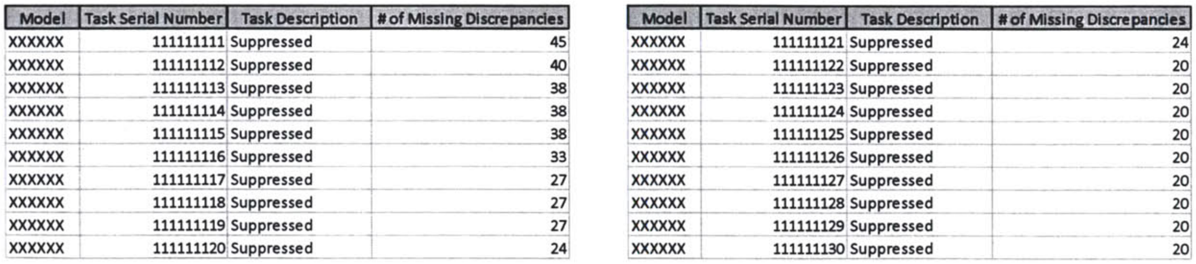
Inspector	Top Issue % of Total	Top 2	Top 3
Inspector A	25%	31%	34%
Inspector B	25%	33%	39%
Inspector C	24%	37%	44%
Inspector D	24%	26%	27%
Inspector E	24%	26%	27%
Inspector F	24%	31%	36%
Inspector G	22%	26%	29%
Inspector H	20%	24%	25%
Inspector I	19%	26%	31%
Inspector J	19%	26%	31%

Table 6: Inspector Top 3 Pet Topics

In this breakdown, the most biased inspectors wrote 20% to 25% of their discrepancies against a single part-defect topic which still represents over triple the amount written against a single topic. This pattern was maintained when the top two and three issues for each inspector were aggregated together. For example, one inspector wrote 44% of his discrepancies against three issues (overall, the largest three issues represented less than 15% of the total). This analysis showed clearly that inspectors have pet issues which biased their output. For more information regarding the utilization of this analysis see section 4.7.

3.6.3 Incomplete work submitted for inspection

Following a conversation with an inspection supervisor, the project team became concerned that some tasks were being submitted to inspection prior to the completion of those tasks. In response, the team attempted to isolate these issues by creating another extension to the primary data analysis. In a preliminary look at the data, the team determined that no common language was used to describe a prematurely submitted task (for example "incomplete" was not a high driver). Instead, the team decided to focus on the "missing" key word and aggregated the data based upon the number of "missing" discrepancies were written for each task on the aircraft. Any task that had five components listed as missing would qualify as a task prematurely submitted for inspection. With this standard, the data illustrated that on average, one task for every shift was submitted prematurely, with a selection of the results included in Table 7 below.



Model	Task Serial Number	Task Description	# of Missing Discrepancies
XXXXXX	111111111	Suppressed	45
XXXXXX	111111112	Suppressed	40
XXXXXX	111111113	Suppressed	38
XXXXXX	111111114	Suppressed	38
XXXXXX	111111115	Suppressed	38
XXXXXX	111111116	Suppressed	33
XXXXXX	111111117	Suppressed	27
XXXXXX	111111118	Suppressed	27
XXXXXX	111111119	Suppressed	27
XXXXXX	111111120	Suppressed	24
XXXXXX	111111121	Suppressed	24
XXXXXX	111111122	Suppressed	20
XXXXXX	111111123	Suppressed	20
XXXXXX	111111124	Suppressed	20
XXXXXX	111111125	Suppressed	20
XXXXXX	111111126	Suppressed	20
XXXXXX	111111127	Suppressed	20
XXXXXX	111111128	Suppressed	20
XXXXXX	111111129	Suppressed	20
XXXXXX	111111130	Suppressed	20

Table 7: Incomplete Tasks Submitted for Inspection

.As can be seen in the chart, some of the tasks submitted early to inspection resulted in the generation of a large number of discrepancies on the aircraft. Each of these tasks would create an unnecessary inspection, wasting resources and potentially causing secondary delays, waiting for inspectors to complete these pointless tasks. Furthermore, when installation returned to actually complete the installation it was frequently more difficult because the job would now occur out of sequence when components from later installations were now in the way. Finally, the efforts to work around these components would also create new and unforeseen opportunities to damage other installations potentially further increasing the amount of rework on the aircraft. For information regarding how this issue was addressed see chapter 4.7.

3.7 General Utilizing the Data

Within Raptor Aerospace, the data proved to be a very effective tool for working with the different departments. Fundamentally, when a strong set of data was present in the conversation, many roadblocks disappeared (discussed in chapter 4.6). The following chapters will provide a more detailed discussion regarding the use of different elements of the data analysis that was completed.

Chapter 4: Working within Raptor Aerospace's Culture

As described in Chapter 1.4, successful change initiatives, including Lean and Six Sigma efforts, need to be tailored to the specific company where they are launched. While there are common ideas and approaches that increase the odds of success, it will be the incorporation of the unique characteristics of Raptor Aerospace that will enable this initiative to succeed. One result of the need for company-specific approaches is that the literature on change initiatives generally focuses on strategic methods (e.g. maintaining an open mind, engaging staff, etc.) without offering much insight into successful tactics. This chapter will first describe the general strategy that the team adopted before offering a discussion of the tactical approaches that were used in this project. Although these tactical approaches may not be applicable for all other companies, they highlight methods that were effective at Raptor and may provide insight for others.

4.1 Strategic Objectives

At the onset of the project, the team identified the following strategies that would govern interactions with other departments and behavioral norms for working within Raptor. The following list provides a summary of these general strategies.

- Communicate frequently and openly with all levels of staff
- Maintain awareness and knowledge of the project,
- Create champions for different aspects of the project
- Listen and consider the ideas and thoughts provided to the team regardless of the source
- Understand the interests and goals of individuals and other departments
- Do not blindside anyone
- When important decisions need to be made, bring everyone that will be affected into the room
- Treat everyone with respect
- Failures are allowed

The team believed that these approaches would assist in increasing the enthusiasm and engagement of other departments enabling a successful effort to motivate a lasting reduction of rework. On several occasions, the team discussed how actions would fit within the above framework. Furthermore, it was believed that this approach could engender a more cooperative and supportive effort for launching Lean and Six Sigma change efforts to the benefit of the firm in the long run. The remainder of this chapter will focus upon specific tactics used throughout this project. It should be apparent how these specific actions fit within the general strategy adopted by the team.

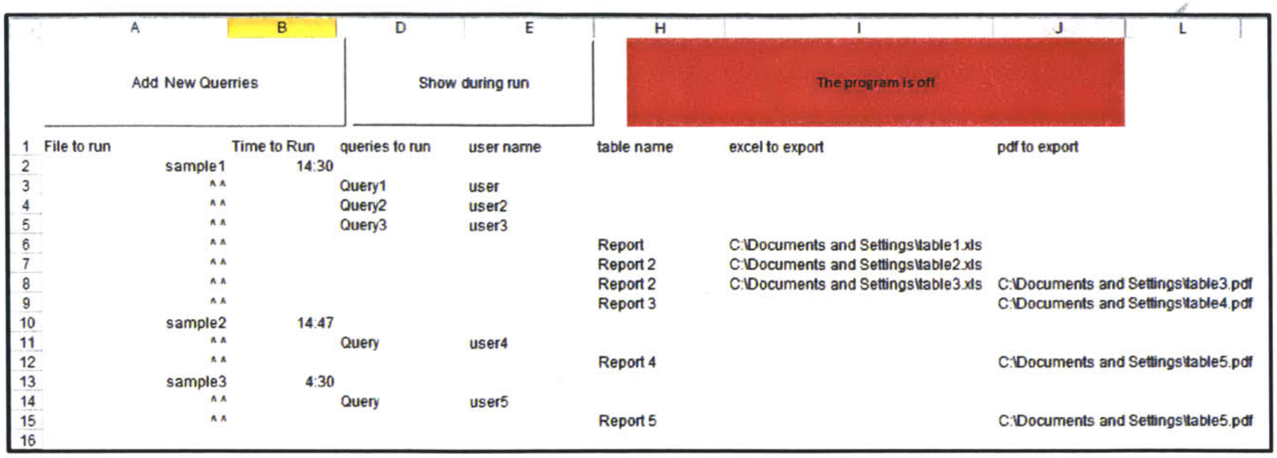
4.2 Routine Informal Information Sharing Meetings

The team conducted periodic meetings with quality, production and engineering managers. In the beginning of the project, these meetings were intended to get feedback on the team's theories and to get input and suggestions for new potential ideas. As the project progressed, these meetings evolved from purely planning into an opportunity to share sensitive information with the departments before releasing it publically, as well as still providing a chance to obtain feedback from these knowledgeable sources. This approach allowed each department an opportunity to solve its own problems or at least to begin to address them early on. Proactive managers were given a chance to turn potentially negative situations into ones that demonstrated good leadership, and showcased their desire to make improvements within the company. Further, no manager would be blindsided, and as a result all were willing to continue working alongside the project team through the entire length of the internship.

In one example, during a meeting with an inspection supervisor, the team learned that some manufacturing supervisors had a habit of submitting jobs to inspection before they were complete. When this occurred, inspectors would be forced to write a series of reworkable discrepancies for every part of the job that was actually incomplete. In some cases, writing these discrepancies occupied an inspector for a full day, longer than it would have taken to complete the installation in the first place. Based upon the discussion, the team developed a modification to the primary data analysis methods to test the validity of the assertion that manufacturing was submitting incomplete jobs. The technical details for this data analysis have been discussed in detail in chapter 3.6.3. This analysis confirmed that, in fact, some jobs had been submitted prematurely for inspection. With the data in hand, the project team approached the line managers for each implicated line in order to discuss potential solutions. Managers then were able to develop their own solutions to the issue before it was released publicly. Overall, the approach enabled the production managers to save face and maintain their integrity, while also demonstrating to the inspection supervisor that his ideas were not only listened to but appreciated and pursued. This approach changed a situation that was potentially damaging into one where everyone benefitted.

4.3 Assisting Other Departments

One of the approaches that the team used to gain support from other departments was to offer assistance in addressing their problems, even if unrelated to the rework project. The thinking behind this approach was twofold. First, it was hoped, that the other departments would be willing to reciprocate the help they received when the project team needed assistance. Secondly, by helping to address the biggest issues, the team hoped to free time for these other groups to work on other projects, time which could be used when requested by the project team. In particular, the team's experience with visual basic for applications was used to help automate common tasks that had become a nuisance to other departments. In one example, the team helped the industrial engineering department automate a series of reports that they assembled on a daily basis by creating a flexible tool for accessing Raptor's database, creating a finished report and placing it in the appropriate place on the share drive. The program was developed with flexibility in mind, and any or all of its features could be incorporated into the generation of any particular report, allowing it to be used on future reports as well as ones that were already being prepared. A copy of the program's interface is included as Figure 14 below. A copy of the code, modified to hide proprietary data, is included in Appendix D along with a full size image of the programs interface.



	A	B	D	E	H	I	J	L
1	File to run	Time to Run	queries to run	user name	table name	excel to export	pdf to export	
2	sample1	14:30						
3	^ ^		Query1	user				
4	^ ^		Query2	user2				
5	^ ^		Query3	user3				
6	^ ^				Report	C:\Documents and Settings\table1.xls		
7	^ ^				Report 2	C:\Documents and Settings\table2.xls		
8	^ ^				Report 2	C:\Documents and Settings\table3.xls	C:\Documents and Settings\table3.pdf	
9	^ ^				Report 3		C:\Documents and Settings\table4.pdf	
10	sample2	14:47						
11	^ ^		Query	user4				
12	^ ^				Report 4		C:\Documents and Settings\table5.pdf	
13	sample3	4:30						
14	^ ^		Query	user5				
15	^ ^				Report 5		C:\Documents and Settings\table5.pdf	
16								

Figure 14: Automated Database Tool

Since the program has been released, it has been shared with several departments in addition to the industrial engineering group. To date it has been used to automate over 20 different tasks and has created significant time savings for Raptor, estimated to be over 30 man-hours a week. As more tasks are added to the program, it is anticipating that it will ultimately provide over 40 man-hours a week.

4.4 Follow up with individuals who provide suggestions and ideas

Sometimes, these efforts needed to be balanced against other priorities of the team. For example, in the case described in section 4.2, an inspection supervisor raised the concern that manufacturing regularly submitted tasks to inspection before the work had actually been completed. During that meeting, the team promised to look into the issue further and validated the claim through data analysis, as described in section 3.6.3. At this point, the team was presented with a conflict in the two objectives, on one hand it was important that the manufacturing managers were not blindsided, but on the other it was important that the team provide follow up with the inspection supervisor, so that he knew his thoughts and ideas had been utilized. However, there was a risk that if the supervisor was told of the results of the data analysis, it might become part of the negative interactions between the departments, inadvertently dragging the rework reduction team into interdepartmental conflicts, potentially impairing the team's ability to work cooperatively and effectively with these groups. Ultimately, the team gave the manufacturing managers time to address the issue before following up with the inspection supervisor and letting him know that not only had his idea been pursued but it also had inspired changes within manufacturing.

4.5 Sharing Credit

One of the most effective tools that the team used was to spread credit for successful projects broadly, emphasizing the assistance that other departments and individuals provided.

Once the team had decided that a specification change was the appropriate fix, the manufacturing engineering department agreed to take primary responsibility for driving the change to a new and approved specification. Instead, over the next two weeks the project stalled as neither design manufacturing, mechanical engineering nor the specification departments were brought into the conversation. Ultimately, the original project leaders took responsibility and drove the project to completion, with manufacturing engineering only providing contact names and a presence during these meetings. Although, the team felt let down by the contribution from manufacturing engineering, publically they chose to highlight manufacturing engineering's contributions and assistance working with the other engineering departments. The thinking behind this approach was that the team would need manufacturing engineering's assistance for other parts of the project in the future and this would help motivate them to contribute.

4.6 Data Driven Approach

One of the most important aspects of the project team's approach was a strong focus on the data, using it to settle open questions whenever possible. As mentioned in section 2.2.5, the departments within Raptor Aerospace frequently pointed the finger of blame at each other whenever problems were identified. If the project team was going to successfully maintain a working relationship with all of these groups, it would be critical that the team maintain a reputation as impartial and fair. If any of the departments believed that the project team was favoring another group, they might stop contributing to the project impairing efforts to reduce rework. To this end, the team used the data and data analysis to settle open questions and as a starting point for opinions that were unpopular with one of the departments. When the team could present these opinions as coming from the data, they were viewed as factual and taken seriously instead of being viewed as finger pointing by another department that was trying to shield itself and spread blame. This built credibility within several of the departments that the project team didn't simply repeat specious claims offered by other groups but instead only chased real issues. As a result, when data was present in the conversation, many of the finger pointing road blocks disappeared allowing more productive and effective conversations to occur.

4.7 Pursuing Bad Behavior

As discussed in section 4.2, members of the project team held informal meetings with numerous stakeholders during the completion of this endeavor. On numerous occasions, these meetings led to the identification of numerous behaviors that were contributing to the amount of rework at Raptor (usually in the form of complaints). While these specific behaviors did not fit into the formal structures that will be discussed in chapter 5, the project team chose to purse these as side issues which could impact the overall performance of the facility. Two examples of particular note were concerns about inspector bias and submission of incomplete tasks. However, given that these topics were raised as complaints in an informal meeting, the team believed it necessary to prove or disprove the behavior through analysis of the data.

In the case of inspector bias, the data analysis, documented in section 3.6.2 provided strong evidence that some inspectors had pet topics that they flagged disproportionally. Although the presence of numerous inspectors on each aircraft would help to offset this effect, this bias could increase the amount of Type I and Type II errors at Raptor. With the data prepared, the project team met with the inspection managers to discuss causes, ramifications and solutions to prejudiced inspection. Although the exact reasons for bias are difficult to determine, it was suggested that one source for this behavior related to the training protocol described in section 2.2.2. Because most inspectors learned how to complete their job through mentorship, these inspectors would exhibit the same tendencies as their trainers propagating bad habits.

The group discussed several potential remedies including rotating inspector teams, modifying the mentorship training program and corrective intervention with specific inspectors. Ultimately, responsibility for addressing the issue was passed to the inspection department while the project team would monitor the data. Once corrective actions had a chance to be implemented, the underlying data would be presented publically to the other departments and upper management.

For the other example, early submission of tasks, the data analysis, section 3.6.3, demonstrated that on average one task every shift was submitted as complete despite strong evidence that work on these jobs was incomplete. Similar to the previous case, the project team prepared the data and organized meetings with managers from production to discuss the causes, ramifications and solutions. One leading cause was pressure on production supervisors to maintain schedule. If a scheduled line roll was approaching and an aircraft only had one task waiting to be completed, the supervisor for that position would want to finish the job and enable an on-time roll of the aircraft. However, if that job were not complete, the supervisor might instruct his crew to submit the job anyways, in order to conceal the delay. Furthermore, the supervisor's decision would not be discovered by management as the resultant discrepancies would not be flagged or identified as unusual. Based upon this source, solutions discussed in these meetings focused around increasing attention on the issue and creating an active monitoring system. In this way, managers could take a more proactive approach to the problem. Like the biased inspector situation, responsibility for the issue was passed to the production department with the project team taking responsibility for monitoring progress. Again, once corrective actions had a chance to be implemented, the underlying data would be presented publically.

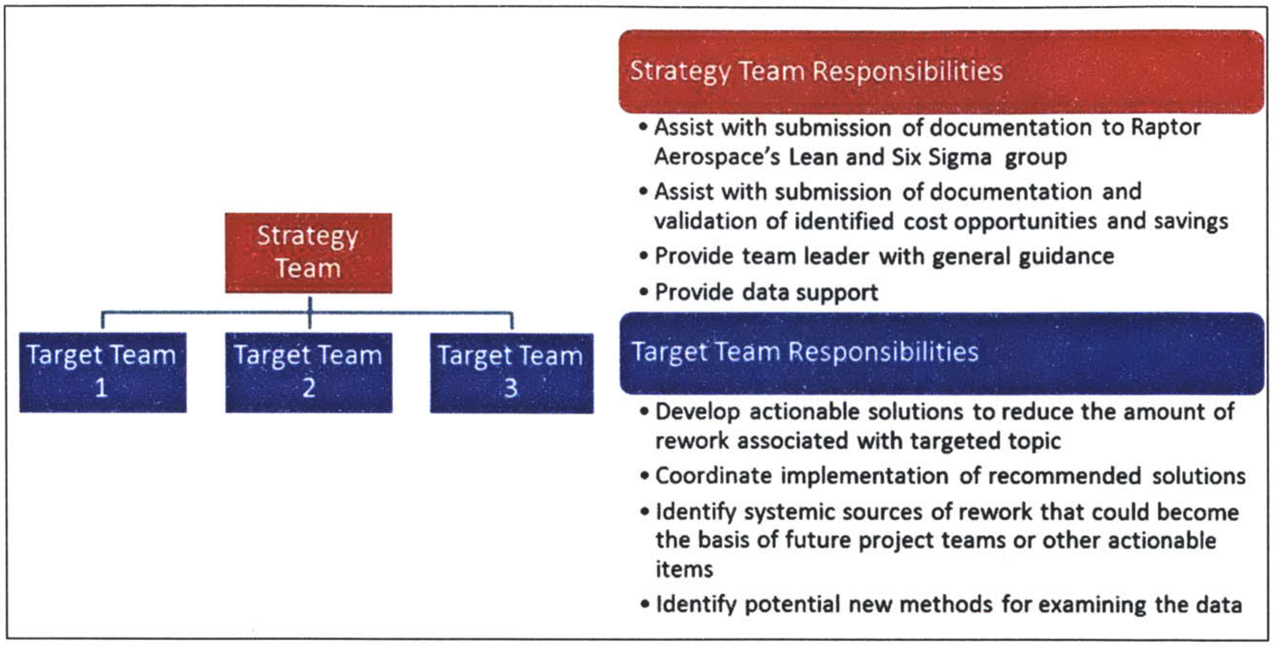
Chapter 5: Project Team Organization,

5.1 Small Project Team Structure

Based upon the primary data analysis results, the project team decided to split responsibilities for each identified topic onto its own targeted team, focusing on the specific top driving topic that was assigned from the data analysis. As Peikert et al. demonstrated, small teams targeting specific operations can be effective in large complex organizations, where modeling the entire operation can be expensive or difficult (1998). Crute et al. also found this approach to be effective in the aerospace industry (2003) although the targeted operations were larger than those of Peikert or that would be used at Raptor.

In addition to the target teams, the project team would be expanded to form a strategy team that would assist and coordinate the new group of target teams (e.g. a lean expert was added to assist with root-cause and corrective action efforts when needed). For convenience, when this thesis refers to this larger group, it will use the nomenclature "strategy team," while the "core team" or "project team" terms will still refer to the original project team responsible for all of the rework reduction efforts. With the creation of the strategy team and the first target teams, general structure and responsibilities for each group were established as displayed in Figure 15.

Each target team would be guided by a leader during the root cause and corrective action phase of their topic. Once this phase was complete, the leader would become the champion for the team's recommendations with ongoing support from the strategy team. However, on the first two teams, it was decided that this responsibility would be shared between two co-leaders. Furthermore, the two project leaders (the author of this thesis and the inspection specialist) were selected for both teams 1 and 2. These two decisions were intended to create consistency within the team structure, which would enable the generation of a template for future team leaders. At the same time, the strategy team would have a longer-term relationship with these leaders enabling the strategy team to learn how to best assist future projects.



Strategy Team

Target Team 1

Target Team 2

Target Team 3

Strategy Team Responsibilities

- Assist with submission of documentation to Raptor Aerospace's Lean and Six Sigma group
- Assist with submission of documentation and validation of identified cost opportunities and savings
- Provide team leader with general guidance
- Provide data support

Target Team Responsibilities

- Develop actionable solutions to reduce the amount of rework associated with targeted topic
- Coordinate implementation of recommended solutions
- Identify systemic sources of rework that could become the basis of future project teams or other actionable items
- Identify potential new methods for examining the data

Figure 15: Team Organization

A key motivation for this approach was the belief that it would enable multiple target teams to be launched simultaneously, while freeing up capacity of the project team to identify potential other opportunities (which would become the target of future teams) and to identify and address systemic rework sources which had a scope that was too large for a small group to address. For more information on an example of a systemic opportunity that was targeted by the core team see Chapter 8.

At the time when this thesis was prepared, a total of seven target teams have been launched and three of them have been brought to completion. The learning's from all of these teams will be summarized in Section 5.4. The action items that came out of the first two teams will be addressed separately in Chapters 6 and 7.

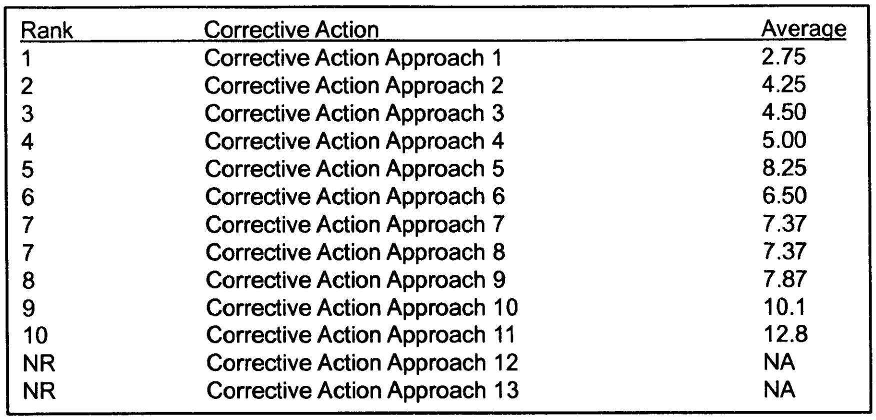
5.2 Target Team Structure

The primary phase of each team would consist of bi-daily meetings for a period of two weeks followed by a week of daily meetings. Additionally, on the first two Wednesdays of the project, the team would only have one meeting to enable the team leader to catch up with preparing documentation for meetings. This format was found to be necessary because on the first two teams, the team leaders wanted to respond to all of the team's questions and needed the extra time to prepare. The first week of these meetings would focus on root cause analysis. During this phase, corrective actions would be gently discouraged. Another finding of these early teams was a tendency for members to switch to corrective actions as soon as a problem was identified. However, when the team jumped into corrective actions too soon, it became more difficult to return to root cause analysis. Further, until all of the root causes were identified, the solutions identified lacked an understanding of the entire picture. This became very apparent in the first team with its early focus on supplier quality as will be discussed in detail in chapter 6.

The root cause phase of the team would focus on driving conversation to identify underlying causes of rework. The only rule for these discussions was that human error was not an acceptable way to explain rework. Our attitude is supported by the work Vaughen and Muschara performed investigating root cause analysis in incident investigation. Vaughen and Muschara emphasized the value of pushing past the human error explanation to get to a problem that might enable a more effective system-related solution (2011). With this basis, where a target team at Raptor identified human error as a root cause, the team leader would be tasked with redirecting the conversation to answer the question of why was human error possible. Root cause analysis would focus on driving to the underlying cause, and takes practice to get right. For example, in one of the teams, the group identified poor quality parts from the supplier as a root cause. However, when the issue was probed further by the team leader, it was discovered that the part was in conformance with the supplier's specification but not the specification being used by Raptor's employees in house, and thus the true root cause was identified.

The second week of the team meetings would focus on corrective actions. This phase would start with the group offering every potential solution they could identify. At this point, solutions would not be limited based upon feasibility, cost or any other consideration and all ideas were encouraged. After all the ideas were exhausted, the team would return to each of the ideas with a focus on identifying actionable solutions. For example, each of the target teams identified training as an underlying cause of the rework and recommended a new training course. Although, this was incorporated on a small scale into the implementation of each project team's recommendations, a large scale training solution was not implemented, due to the reasons discussed in chapter 2. Instead, the team recorded the recommendation and is building a record of the request for a future initiative.

Finally, once the team felt comfortable with a large set of potential corrective actions, they would be divided into different functional categories (e.g. from team 1, wedge requirements, multiple clamp setups, etc.). Next, the team would rank ideas in each category based upon value and effectively. Figure 16 offers a sample ranking provided by one of the teams.



Rank	Corrective Action	Average
1	Corrective Action Approach 1	2.75
2	Corrective Action Approach 2	4.25
3	Corrective Action Approach 3	4.50
4	Corrective Action Approach 4	5.00
5	Corrective Action Approach 5	8.25
6	Corrective Action Approach 6	6.50
7	Corrective Action Approach 7	7.37
7	Corrective Action Approach 8	7.37
8	Corrective Action Approach 9	7.87
9	Corrective Action Approach 10	10.1
10	Corrective Action Approach 11	12.8
NR	Corrective Action Approach 12	NA
NR	Corrective Action Approach 13	NA

Figure 16: Sample Corrective Action Ranking

The last two ideas in Figure 16 were ultimately determined by the team to be of no value and their actual scores were replaced with an "NA." In this scale, a one would indicate the most important idea with each idea thereafter being scored accordingly. Finally, the team reviewed the rankings of the ideas to see if they wanted to make any switches to the order based upon conversation. As can be seen in Figure 16, the team decided that the fourth corrective action approach was more valuable than its score originally indicated, and moved it higher up the priority list. An interesting lesson that arose from this ranking was that in general while there was a preferred topic, there still was a fair variety of opinion about the effectiveness of each idea. In fact, in the example provided, the best idea only ranked an average score of 2.75 and after that, the numbers increased into the 4-5 range.

5.3 Common Tools for each Target Team

Prior to each project team launch, each target team was provided with its own folder on the shared drive, outfitted with a common set of documents and folders that had proven useful in the operation of previous teams. These common tools included the following:

- **Detailed Targeted Analysis**: Each team was provided with a series of breakdowns of the data that might prove helpful to the team, including a heat map of target defects overlaid on an aircraft, a breakdown of the specific tasks which create the most rework, a time history of the target topic, and a copy of the underlying data..
- **Root Cause and Corrective Action Tools**: Each team was provided with a set of tools to facilitate the root cause and corrective action phases of target teams including fishbone diagrams, five why sheets, corrective action template and more proprietary documents from Raptor's lean organization. Unfortunately, none of these documents could be included due to their proprietary nature. Figure 17 is a sample of a fishbone diagram taken from Vassilakis and Besseris' work at a large aerospace company's maintenance division, identifying causes that led to a poor nozzle assembly process (2009). A larger copy of this figure is included in Appendix A.
- **General Guidance Tools:** Each team was also provided with team organizational forms and documents to be submitted to get credit from Raptor's lean group and cost recording system. In this way, each project leader would be credited with the dollar savings and the team would be credited for participating in a lean initiative. The Strategy team also assisted with the completion of these forms structuring periodic check-ins. These check-ins provided a mechanism for monitoring progress.

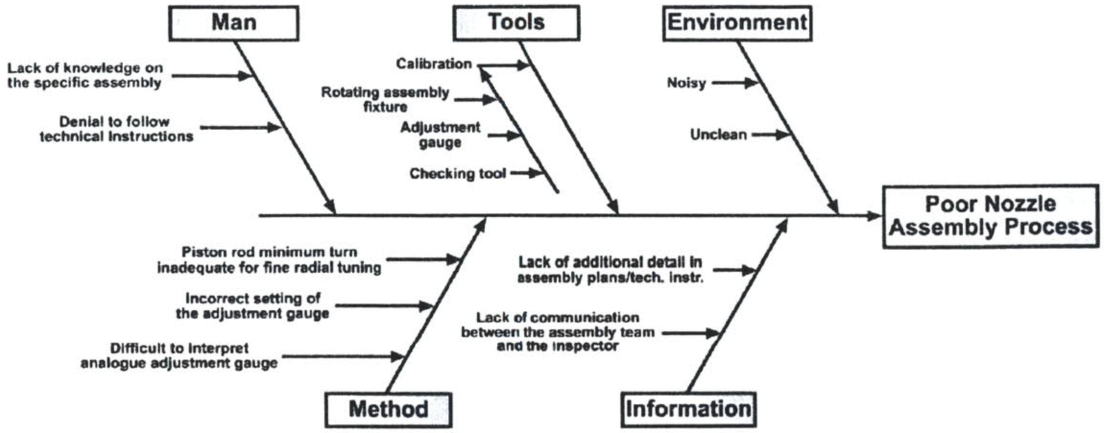

Man
Tools
Environment
Lack of knowledge on the specific assembly
Denial to follow technical instructions
Calibration
Rotating assembly fixture
Adjustment gauge
Checking tool
Noisy
Unclean
Poor Nozzle Assembly Process
Piston rod minimum turn inadequate for fine radial tuning
Incorrect setting of the adjustment gauge
Difficult to interpret analogue adjustment gauge
Lack of additional detail in assembly plans/tech. instr.
Lack of communication between the assembly team and the inspector
Method
Information

Figure 17: Sample Fishbone Diagram (Vassilakis and Besseris, 2009)

5.4 Lessons from Teams 1-3

One of the key objectives of all of the target teams was to provide lessons and feedback which would enable future teams to learn from their mistakes and achieve higher levels of success. This objective proved particularly relevant for the third team, which failed to produce any actionable solutions, but did provide a wealth of knowledge for improving the team format for the future. The remainder of this section will be broken into two parts, the first focusing on teams 1 and 2, while the second will focus on the lessons from team 3.

5.4.1 Teams 1 and 2

One of the key lessons learned from the first two teams was the importance of selecting installers and inspectors who had appropriate work experience, were interested in making improvements, and were willing to participate in a give-and-take approach. Additionally, if these criteria were not met, it was important to substitute a new person into the team, so that one individual would not poison the results. When all of the criteria were met, the team engaged in very productive conversations, drilling down to actionable root causes and effective solutions. On the other hand, if any of the three prerequisites were missing, the team would cease to make progress. For example, on the first team, one of the original installers knew the topic exceptionally well and wanted to fix the issue. However, this individual was convinced that he knew the true cause of the problem, and believed that conversation about other factors was a waste of time. As a result, whenever the leaders attempted to discuss other causes of problems, he would interject and kill the discussion. Ultimately, it was only when this individual was replaced with another installer that a true root cause analysis was completed. More importantly, the team later discovered that this installer had been focused on a symptom and not on the true source of the problem.

Another important lesson concerned providing team members with enough time between meetings to gather data supporting their arguments, or for the strategic team to answer questions raised. If the meetings were held too frequently, this information was not available by the next meeting, and there was a tendency for the team to harp on the question rather than move on to more productive topics. It was for this reason that the strategic team decided to meet only once during the first two Wednesdays.

5.4.2 Team 3

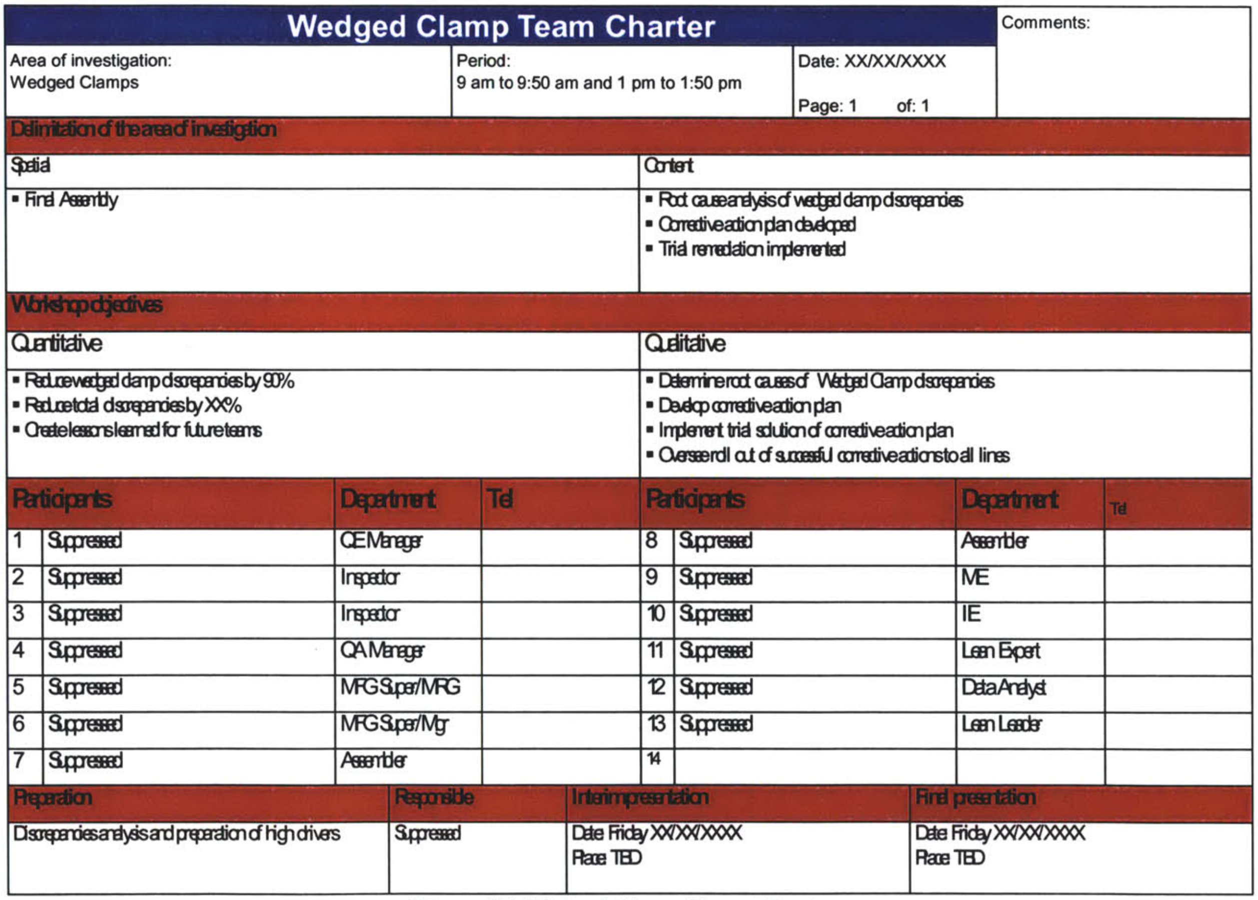
In order to spread the rework reduction efforts to other locations, the strategy team decided to launch team 3 in a sister facility, which required a day's travel to visit. The team was led and composed of members from the sister site, with the strategy team participating via video and phone conference in all of the meetings. Early in the corrective action phase of this team, it became apparent that the presence of the strategy team was a distraction. When everyone is in the same room, each member was given an equal opportunity to contribute and be heard. However, when a person was participating via the phone or a video conference, the group inherently offered the remote individual(s) an unwarranted degree of attention. As a result, every time a member of the strategy team spoke, the entire team would stop their conversation and listen, destroying the flow of discussion and losing many of the ideas being presented. From this experience, the strategy team decided that in future sister-site projects, the strategy team would limit its interactions to working only with the project leader, in order not to become a distraction.

A second issue that was identified from the third team was the need to prepare the team leader for the general flow of the root cause and corrective action meetings. The leader for this team had experience facilitating this type of effort, but did not realize the need to drive back to actionable solutions. As a result, the team became focused on systemic issues that were worth pursuing but require a substantial investment of money and time that Raptor was not prepared to deliver. As a result, instead of creating ideas that generated momentum which could eventually be used to tackle larger issues, the team only developed solutions that were too large in scope to be tackled with the available resources. From this experience, the strategy team learned the importance of communicating the goals for the target team frequently and clearly.

Chapter 6: Wedged Clamp Target Team

6.1 Team Goal and Composition

The first target team was launched to address the issue of improperly wedged clamps that was identified as one of the high drivers by the primary data analysis phase. The team for this project was built with support from every department within final assembly, as can be seen in the completed Project Team launch form (key information has been disguised) included as Figure 18. A full size copy of this charter is included as Figure 32 in Appendix A.



Wedged Clamp Team Charter

Comments:

Area of investigation: Wedged Clamps	Period: 9 am to 9:50 am and 1 pm to 1:50 pm	Date: XX/XX/XXXX Page: 1 of: 1

Delimitation of the area of investigation

Spatial	Content
▪ Final Assembly	▪ Root cause analysis of wedged clamp discrepancies ▪ Corrective action plan developed ▪ Trial remediation implemented

Workshop objectives

Quantitative	Qualitative
▪ Reduce wedged clamp discrepancies by 90% ▪ Reduce total discrepancies by XX% ▪ Create lessons learned for future teams	▪ Determine root causes of Wedged Clamp discrepancies ▪ Develop corrective action plan ▪ Implement trial solution of corrective action plan ▪ Oversee roll out of successful corrective actions to all lines

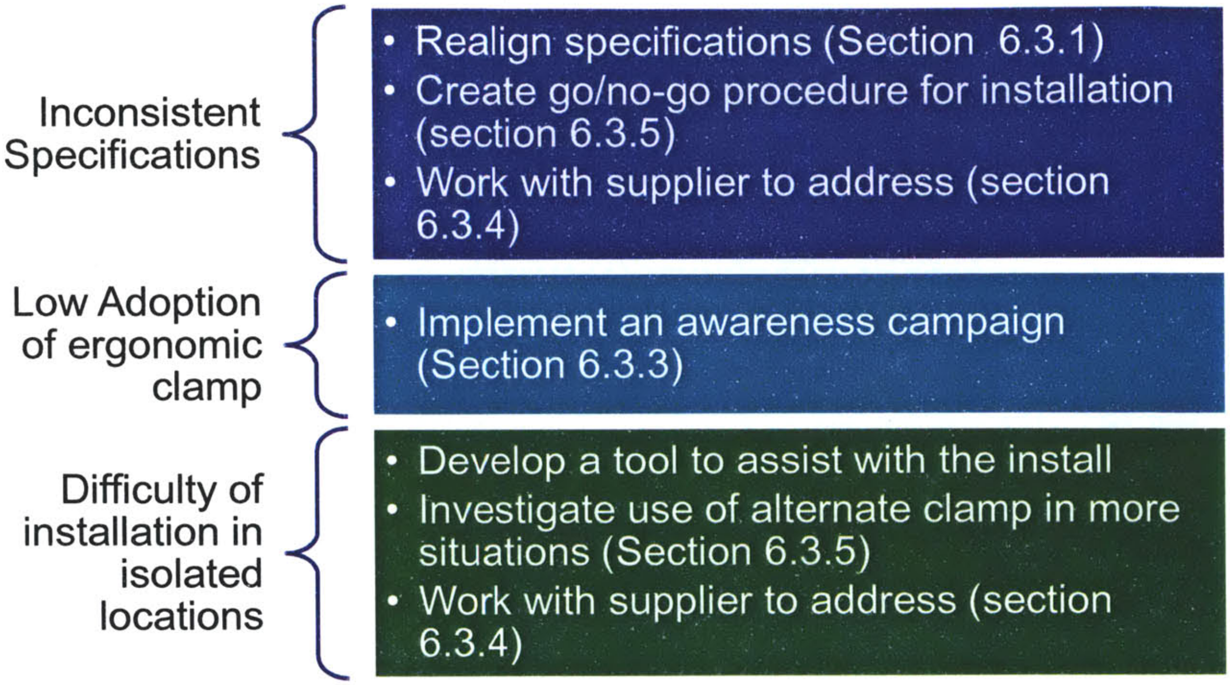
	Participants	Department	Tel		Participants	Department	Tel
1	Suppressed	QE Manager		8	Suppressed	Assembler	
2	Suppressed	Inspector		9	Suppressed	ME	
3	Suppressed	Inspector		10	Suppressed	IE	
4	Suppressed	QA Manager		11	Suppressed	Lean Expert	
5	Suppressed	MFG Super/MRG		12	Suppressed	Data Analyst	
6	Suppressed	MFG Super/Mgr		13	Suppressed	Lean Leader	
7	Suppressed	Assembler		14			

Preparation	Responsible	Interim presentation	Final presentation
Discrepancies analysis and preparation of high drivers	Suppressed	Date Friday XX/XX/XXXX Place TBD	Date Friday XX/XX/XXXX Place TBD

Figure 18: Wedged Clamp Team Charter

6.2 Key Results

As described in Chapter 5, the team conducted a root cause and corrective analysis of the wedged clamp issue, and identified several causes which could be then targeted for repair. Figure 19 summarizes a selection of the root causes and corrective action analysis that was completed, highlighting the issues that the team believed to have the highest impact.



Inconsistent Specifications

- Realign specifications (Section 6.3.1)
- Create go/no-go procedure for installation (section 6.3.5)
- Work with supplier to address (section 6.3.4)

Low Adoption of ergonomic clamp

- Implement an awareness campaign (Section 6.3.3)

Difficulty of installation in isolated locations

- Develop a tool to assist with the install
- Investigate use of alternate clamp in more situations (Section 6.3.5)
- Work with supplier to address (section 6.3.4)

Figure 19: Selection from Wedged Clamp Root Cause and Corrective Action Findings

6.3 Implementation of Key Recommendations

Once the root cause and corrective action phases of the first team were complete, the two project leaders were tasked with seeing the top ideas to completion. In some situations, individual members of the team were brought back into the conversation in order to emphasize some of the key findings of the teams, but for the most part at this point, the bulk of the project shifted to the leaders and the departments that would need to make the recommended changes. The remainder of this section will highlight how these changes were implemented.

6.3.1 Specification Realignment

One of the critical findings of the root cause and corrective action phase of the team was that the specifications given to the supplier and employed within Raptor were inconsistent. It was possible for the supplier to provide a part that was in complete conformance with Raptor's supplier requirements, the governing safety regulators, and the requirements of every one of Raptor's customers, but would still result in Raptor inspectors writing a reworkable discrepancy against its own installers.

In order to rationalize the specifications, the project team (with assistance from manufacturing engineering), coordinated a meeting with all of the parties responsible for creating and managing these disparate documents. This meeting focused on understanding the logic underneath the language choices in each specification, looking to determine if these documents were attempting to accomplish different things or if they were using different methods to accomplish the same goals. In either case, once the group understood why the differences were present, participants would be able to determine the best way to eliminate this variability, while maintaining the highest quality product.

In the case of wedged clamps, it turned out that the specifications were trying to accomplish the same goal but through different means. Once the group reached this conclusion, the conversation flowed naturally into a question of which version of the specification should be used at Raptor and its suppliers (or if a new version should be created). Ultimately, the group settled upon modifying Raptor's specification to match the version given to its suppliers.

After the meeting, the specification drafter began work on a revised document which would be submitted to all of the meeting participants for comments. At the same time, mechanical engineering began notifying individuals who were not present at the meeting but would need to sign the final document (e.g. FAA representatives) and the project team began preparations to roll out the changed specification in final assembly. This roll out plan will be discussed in section 6.3.3.

6.3.2 New Tool

Based on one of the recommendations from the project team, a new tool was designed and prototyped by the tool engineering department at Raptor Aerospace. This tool was then modified based upon feedback from installers and prepared for widespread release. At the time of the writing of the thesis, work on the tool was almost complete and an awareness campaign was being planned. More information regarding the tool cannot be discussed here, due to its proprietary nature.

6.3.3 Training Roll-out

The training program for the new requirements included the posting of two "newsflashes" to the inspection and installation crews, the creation of an instructional video, and a flier to be posted on video monitors throughout final assembly. This training regimen was intended to address both current gaps in the knowledge about the wedge clamp issue, as well as to inform installers and inspectors about other changes being made by the project team. The first newsflash was issued after the root cause and corrective action phases of the primary team were complete, but before any of the corrective actions were implemented. This newsflash focused on increasing awareness of the wedged clamp issue, and the availability of an alternative clamp, while also informing company staff that the topic was being pursued as an improvement opportunity. After the specification change was complete, the next three parts of the training program were rolled out simultaneously to maximize awareness of the new standard.

The instructional video was approximately 10 minutes in length and began with a highly respected manager (and former inspector on the floor) discussing the new standards and why they were created. Following the introduction, the video presented several clamp installations highlighting the features of the new standard and the simplification of the pass/fail criteria. For the roll-out, the video was shown to inspection and manufacturing in each shift followed by a 20 minute question and answer session to discuss any questions that staff had about the new standards. Assisting with the roll-out, the inspection managers made it a priority to ensure that their crews clearly understood these changes, leading to a high level of adoption and a consistent application of the new standards by the inspection department.

6.3.4 Supplier Coordination

Throughout the implementation of the in-house changes, the project team was also working with the clamp suppliers to make changes to the clamp that might enable an easier more efficient installation of the part on the aircraft. In these conversations, the clamp manufacturers had developed several solutions which might be an effective means to reach this goal, but these ideas cannot be discussed in this thesis for proprietary reasons. Work on selecting an improvement was still ongoing when this thesis was prepared.

6.3.5 Abandoned Recommendations

Several of the corrective actions that were identified by the target team ultimately ended up being abandoned by the project team for various reasons. Two prominent examples were the go/no-go marker and the alternate clamp that was examined. In the case of the go/no-go marker, the specification realignment ultimately superseded the goal of the marker, causing it to be abandoned (this was an anticipated outcome of the team). Regarding the alternate clamp, when the idea was reviewed in more detail by engineering (with participation from several engineering groups), it was determined that while the alternate clamp would address one issue, it would create another problem and therefore was ultimately abandoned.

6.4 Long Term Performance

Following the full implementation of the primary corrective actions identified for the "wedged clamp" rework, the team continued to monitor the performance of the issue to determine the effectiveness of the changes. One concern of the team at the beginning of the project was that improvements might be short-lived, with the amount of rework decreasing while the issue was spotlighted, but returning to the previous level as soon as the attention moved to a new topic. Figure 20, shows the amount of "wedged clamp" rework that was created before and after the changes were announced for one line. The roll-out campaign ultimately proved to be highly successful as a sharp immediate decrease in rework followed the release of the roll-out campaign. Highlighting the positive reception of the changes,

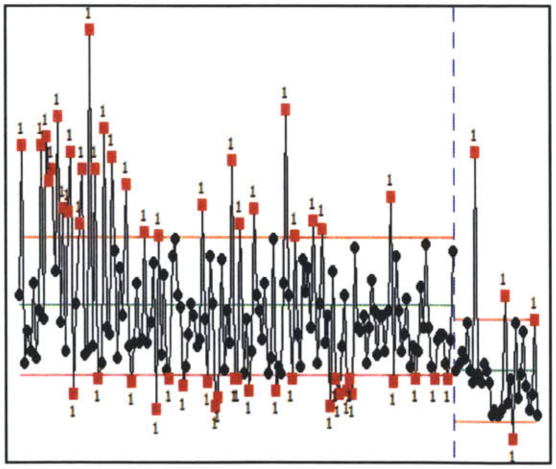


Figure 20: Wedged Clamp Control Chart

during the instructional video presentation to the facilities' second shift, one of the installers raised his hand and commented, "This is great, what took so long?"

Across all of the lines, the long-term performance was monitored, generating the following statistics.

- Immediately after the changes were announced, the amount of "wedged clamp" rework dropped by 40%
- Over time, this number improved and the amount of "wedged clamp" rework dropped an additional 33% bringing the total reduction to 60% of the original amount
- Improvements were achieved in every line, although the exact amounts varied

The data indicates that not only were the corrective actions effective in reducing the amount of "wedged clamp" rework that was done at Raptor, but that these changes were self-sustaining and improved over time as staff shared insights into how to perform the installation and complete inspections. One example of this behavior was reported to the team by a floor supervisor when an installer completed a clamp installation that failed to meet the new criteria. As soon as the inspector saw the installation, he walked over to the installer and asked how it was possible for him to make this mistake. Dragging the installer over to one of the plasma screens displaying a proper clamp installation, the inspector told the installer he never wanted to see a mistake like that again.

To the team, this anecdote highlighted an unanticipated secondary cultural benefit to employees who were not involved in the change effort. Here was an inspector improving communication and accountability by confronting an installer over a poor quality job. If the specifications had remained vague, the installer could argue that his work might conform to the requirements, but with the new instructions it was clear that rework was required. Ultimately, it is likely that this type of conversation on the floor is the primary motivator for the continuing improvements observed over time.

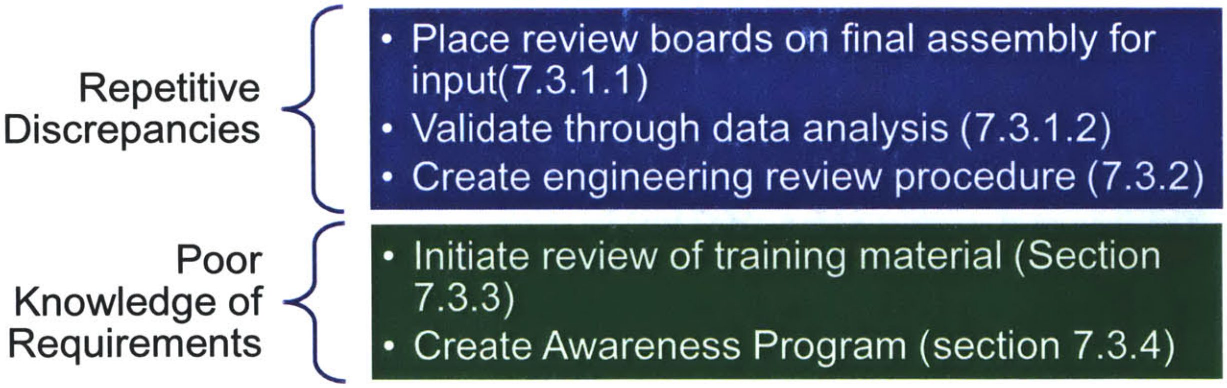
Chapter 7: Chafing Harness Target Team

7.1 Team Goal and Composition

The second target team was launched to address the issue of chafing wiring harnesses, which was identified as one of the high rework drivers by the primary data analysis phase. The team for this project was built with support from every department within final assembly, as can be seen in the completed Project Team launch form (key information has been disguised) included as Figure 33 in Appendix A.

7.2 Key Results

Once the chafing harness team completed its root cause and corrective analysis phases, it identified several causes which could be targeted for repair. Figure 21 summarizes a selection of these topics and proposed solutions, highlighting the issues that the team believed to have the highest impact.



Repetitive Discrepancies

- Place review boards on final assembly for input(7.3.1.1)
- Validate through data analysis (7.3.1.2)
- Create engineering review procedure (7.3.2)

Poor Knowledge of Requirements

- Initiate review of training material (Section 7.3.3)
- Create Awareness Program (section 7.3.4)

Figure 21: Selection from Chafing Harness Root Cause and Corrective Action

7.3 Implementation of Key Recommendations

7.3.1 Repetitive Discrepancies

One of the key findings of the target team was the identification of repetitive discrepancies as a significant source of chafing harness rework. Specifically, the team determined that there were some jobs with specific components that were extremely difficult to install, to the point where they frequently resulted in the exact same type of discrepancy on each aircraft. If a systemic approach to identifying and eliminating these topics could be developed, the team believed that repetitive discrepancies represented an opportunity to gain a meaningful reduction in all types of rework, and not just chafing harness examples. Furthermore, although the exact cause for each of these repetitive issues could be different, the team anticipated similarities between each occurrence, which would enable common solutions. Finally, the team leaders believed that this topic represented an opportunity to create a lean practice with cultural impact at Raptor.

The team identified two primary methods for tackling these topics that would be implemented and tested simultaneously. One approach would be to place review boards on the floor of final assembly to elicit feedback from both installers and inspectors. The other strategy was to review the available data and isolate a means of identifying repetitive issues.

7.3.1.1 Review Boards

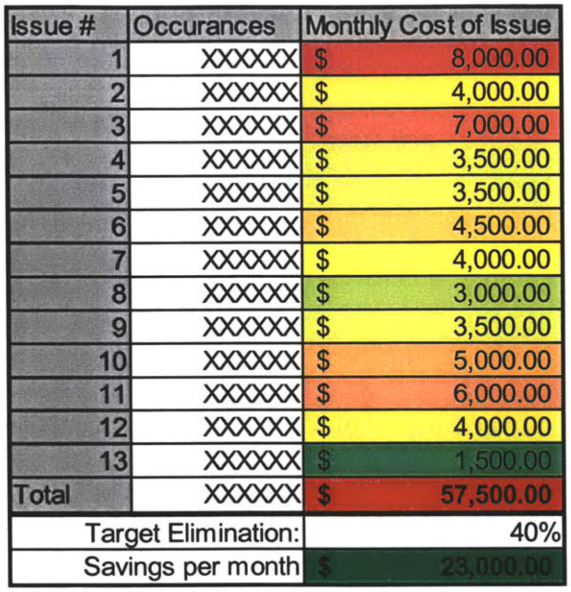
The first approach for repetitive issues was to place review boards on the floor of final assembly to encourage installers and inspectors to identify repetitive issues that could be tackled. The team started with one board which elicited several examples. Additionally, the installer and inspectors who participated on the team also provided examples of the issues that frequently impacted their work. Once approximately a dozen opportunities were identified, the team brought these to engineering for discussion as will be addressed in section 7.3.2.

7.3.1.2 Data Analysis

The second approach that was identified by the team was to perform a data analysis to verify the existence of repetitive issues and also to potentially identify examples. This data analysis is summarized in section 3.6.1, and resulted in a method that both verified the potential size of repetitive issues and also provided a means to identify specific opportunities, based upon the generation of similar discrepancy reports. Once, the team felt confident that verified repetitive defects had been identified, the team created a cost estimate for thirteen of the issues. This cost estimate is included as Figure 22, with the underlying numbers disguised and obscured as appropriate.

7.3.2 Engineering Review of Repetitive Issues

Once the team had developed two methods for identifying repetitive issues, the team sought assistance from engineering to eliminate the causes of these discrepancies, which met with different levels of success. When the ideas from the review boards were presented, engineering was reluctant to pursue them, insisting that manufacturing and inspection must obtain permission from engineering before writing a formal request for an engineering change (note that this in contrast to Raptor's written procedures which state that the request for an engineering change is meant to enable manufacturing and inspection to seek engineering help in areas where they are having trouble). Furthermore, when the team arranged to show engineering the first group of ideas, engineering rejected all but one of these ideas, offering reasons such as the following for rejecting the others.



Issue #	Occurances	Monthly Cost of Issue
1	XXXXXX	$ 8,000.00
2	XXXXXX	$ 4,000.00
3	XXXXXX	$ 7,000.00
4	XXXXXX	$ 3,500.00
5	XXXXXX	$ 3,500.00
6	XXXXXX	$ 4,500.00
7	XXXXXX	$ 4,000.00
8	XXXXXX	$ 3,000.00
9	XXXXXX	$ 3,500.00
10	XXXXXX	$ 5,000.00
11	XXXXXX	$ 6,000.00
12	XXXXXX	$ 4,000.00
13	XXXXXX	$ 1,500.00
Total	XXXXXX	$ 57,500.00
	Target Elimination:	40%
	Savings per month	$ 23,000.00

Figure 22 Cost Breakdown, 13 Repetitive Issues

- This is not a discrepancy and no change is needed.
- This could easily be done another way that conforms to the specifications.
- There is nothing that can be done about this one.

In most of these cases, the installers and inspectors who were familiar with the job insisted that engineering was mistaken and that these were in fact repeat problems. For instance, in one case, both inspection and installation agreed with engineering that a particular job was completed correctly, but it looked like a poor installation that would be consistently rejected by the customer, which explained the reworkable discrepancy. At this point, the team was concerned that the low success rate for this approach would discourage ongoing participation from the groups within manufacturing, and ultimately might turn into a source of tension if each group blamed the others for the failure.

While the efforts above were ongoing, the team separately brought the results of the data analysis to engineering, including the savings estimate for the first thirteen issues. Unlike the review board experience, these issues would not have an installer or inspector who could point the engineer directly to the problem, but would instead require engineering to investigate the issue independently to determine the exact cause. Now, one department would be responsible for all aspects of the repetitive discrepancy issue, having to identify the problems, create solutions, and implement changes in the design. Furthermore, the cause of the problem would not be important (and hence finger-pointing would not threaten the project), but would instead be interpreted as part of the current state. As a result, the team convinced engineering to take responsibility (and credit) for removing these problems.

With engineering motivated to pursue topics that were identified through the data, but resistant to tackling the issues being presented by the installers and inspectors (when it was unclear which department would get credit for any achieved savings), the team decided to put the review boards on hold. If the team was successful with the data-driven approach, establishing momentum for removing repeat issues, it could then revisit the review boards for additional input. At this point, weekly meetings were established for engineering to present its efforts to reduce repeat issues where engineering would take primary responsibility for carrying out the remainder of this part of the project.

7.3.3 Review of Training Material

One of the key findings from the target team was that the requirements for chafing harnesses were unclear and poorly understood by both the installation and inspection crews. The team arranged a kaizen event to examine the available training literature at Raptor, to assess the effectiveness of the material and develop a plan to address the insufficient knowledge on the floor of final assembly. During this review of the literature, a number of factors were identified, which contributed to the poor understanding of the requirements to prevent a chafing harness.

First, the group identified that the training regimen was insufficient. As discussed in Chapter 2.2.3, the kaizen event found that both the new-hire training and routine refresher training were lacking. In particular, the lack of refresher training resulted in most installers and inspectors having a poor understanding of the material and no formal means of sharing best practices. For example, the literature contained advice for installing harnesses to prevent chafing that not a single installer or inspector in the room recognized as having seen before. Next the group found that the literature was difficult to work with, being stored in several lengthy documents of up to several hundred pages. At this length, it was unlikely that this documentation would maintain the attention of the typical worker. Finally, the team identified several suggestions within the literature which had been rendered obsolete by changes in the specifications. This advice would need to be eliminated from any formal training package in order to be effective. Once the kaizen event was finished, the team had created a series of recommendations for a training protocol with regard to the chafing harness issue.

To address the issue, the team approached the training department within Raptor Aerospace to create a program based upon the recommendations from the kaizen event. However, the training department stated that the remainder of their yearly budget was already committed, and they would not be able to assist with the project for three months, which was after this author would have departed the site. Furthermore, the team believed that preparing a training program without the training department might have long term negative ramifications that would outweigh the benefit of the program. Additionally, when the team approached the managers from installation and inspection, both groups expressed reluctance to adding a regular training protocol, due to fear that it would negatively impact their ability to meet the schedule. Ultimately, the team decided to table the initiative, until it felt that it had built up enough momentum with other efforts to improve the likelihood of success.

7.3.4 Awareness Program

As of the writing of this thesis, the team was in the progress of preparing an awareness campaign, similar to the one created by the wedged clamp team, although with a higher focus on general knowledge and advice for installers and inspectors. Due to the preliminary nature of this awareness campaign, the specific details are not included in this thesis.

7.4 Team Impact on Reworkable Discrepancies

When the author left the site, engineering had successfully developed solutions for several of the top repetitive issues and was in the process of implementing these ideas. These solutions included relocating parts, creating new drawings to assist installation, increasing the availability of tools, and design changes. The overall progress of these changes was monitored by the project team to determine the effectiveness of solutions. Specific results of these efforts are not included in this thesis for proprietary reasons.

Chapter 8: Systemic Rework Solution, Automated Rework Metric

As was mentioned in Chapter 5, the strategy team was tasked with developing systemic approaches for the reduction of rework, in addition to assisting the target teams that were formed following the primary data analysis. Through a series of discussions with both production and inspection managers, it was brought to the team's attention that no long-term quality performance metrics were in use at the facility. In fact, the only quality tool in use by production was a rework assignment tool, which enabled manufacturing to assign rework back to the installer who created it (this was only possible for approximately 50% of rework). However, while this tool informed installers of their current outstanding rework and created incentives to have the same individual make the repair, it did not provide any means to monitor their long-term performance or to determine if overall quality on the floor was improving or getting worse. Thus, the first systemic approach that the core team adopted was to create a time-history metric for the manufacturing floor. Once a tool to create the metric was completed and demonstrated to all levels of manufacturing, it would be handed to the industrial engineering group for permanent operation.

8.1 The Metric Tool's Requirements

The quality metric tool that would be developed needed to be highly automated to minimize the operator's commitment once the program was handed to another department. The tool would also need to have a simple and easy to understand interface to facilitate this transfer. As a result, it was required to have a maximum of one minute set-up time, a maximum of twenty minute computer run time and a clear interface. Additionally, to be effective, the tool would need to generate appropriate reports for every level of manufacturing. These reports would provide the number of discrepancies created in the twenty most recent times a task was completed. This scheme allows each level of manufacturing to have productive and clear communication points to discuss with both their superiors and subordinates. For example, the site director would know the tasks producing the most discrepancies in the facility in order to discuss these jobs and motivate the line managers to improve their performance. Figure 23 outlines the reports that would be delivered to each level of the manufacturing hierarchy, illustrating the communication points that would be created.

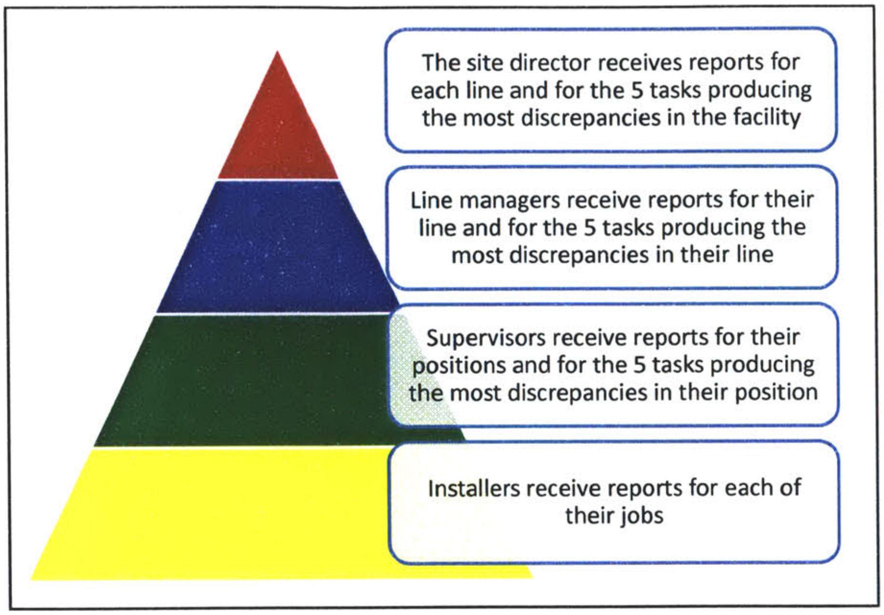
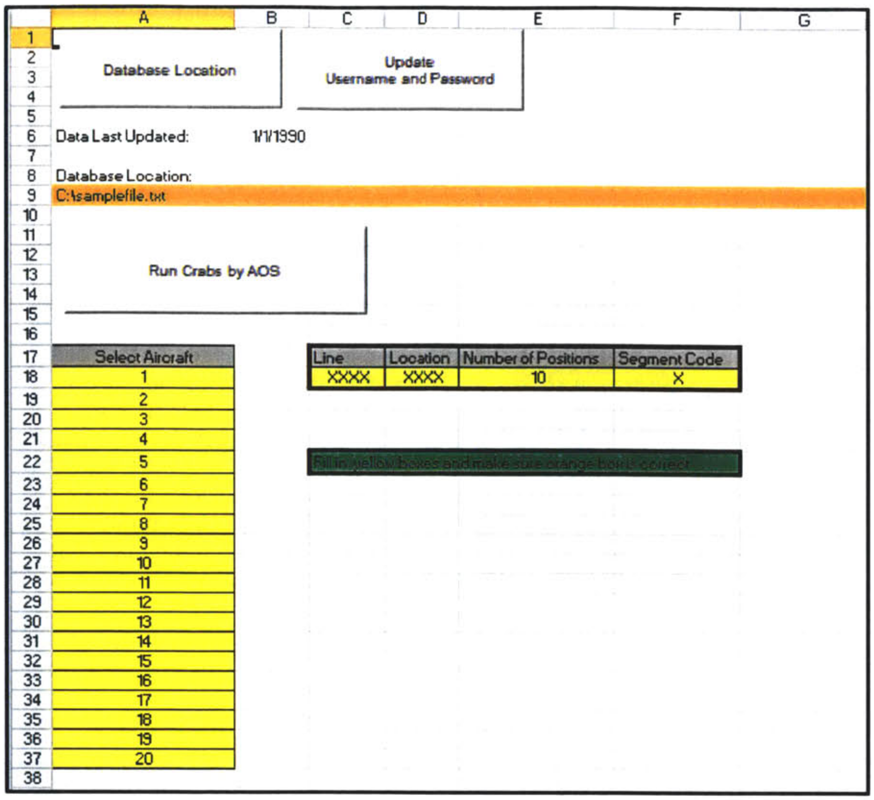

The site director receives reports for each line and for the 5 tasks producing the most discrepancies in the facility
Line managers receive reports for their line and for the 5 tasks producing the most discrepancies in their line
Supervisors receive reports for their positions and for the 5 tasks producing the most discrepancies in their position
Installers receive reports for each of their jobs

Figure 23: Reports for Each Level of Manufacturing

One final output of the report, which is not included in the above hierarchy, was a report of all tasks performed (separately for each line) which had not received a discrepancy against them in the last 20 aircraft (number disguised). This report would be posted in front of each position in the line, publically acclaiming installers who demonstrated long-term quality performance.

8.2 Tool's Interface and Construction

As mentioned in the requirements, the program's interface was critical for the successful implementation of the data metric. The tool would need to be clear and easy to operate in order to ensure that it would not be abandoned early in the process. The program's interface was thus designed to be clean and simple to facilitate the transfer of responsibility to the Industrial Engineering group. Based on this expectation, the interface was designed to have a front page based in Microsoft Excel. The back end of the program would utilize a combination of VBA functionality, javascript, and utilization of other commercial software to provide all of its capabilities. A copy of the tools interface is included as Figure 24.



Data Last Updated: 1/1/1990

Database Location:
C:\samplefile.txt

Select Aircraft
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

Line	Location	Number of Positions	Segment Code
XXXX	XXXX	10	X

Fill in yellow boxes and make sure orange box is correct

Figure 24: Automated Discrepancy Report Interface

8.3 Sample Reports

Figure 25 is a sample of one of the reports that was generated by the report.

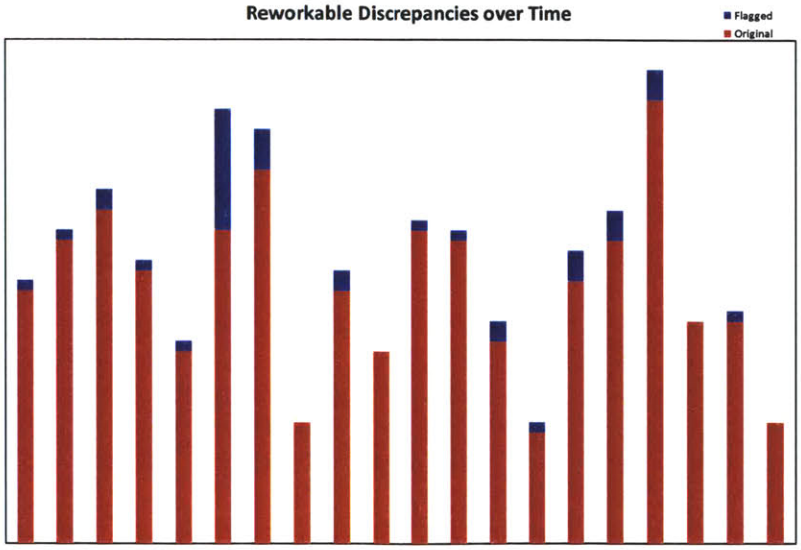


Figure 25: Sample Discrepancy Tool Report

This figure represents one of the summation charts totaling up the performance of a group of tasks that would be prepared for a supervisor or manager. Each red bar indicates the total number of discrepancies generated. The program also "flags" certain discrepancies which met select conditions, indicating that it required more work to repair than a typical discrepancy. These "flagged" discrepancies are added onto the chart in blue and, were intended to draw attention to a negative behavior that was creating an additional expense. The team hoped that by highlighting this behavior, the metric could help to eliminate it, as well as to reduce the total number of discrepancies.

Although not shown in any of the sample charts that will be provided, the program creates an automatically adjustable target line for each of the summation charts generated, allowing each position to see how it is performing compared to its goals.

8.4 Examples

During the preliminary beta testing of the metric, numerous groups within final assembly identified potential uses and value from the product. This section will highlight three examples that demonstrate some of the value that was realized even before a complete launch of the program.

8.4.1 High variability in a single task

In one example, one of the line managers was reviewing all of the reports for his line and identified Figure 26 for further investigation. This chart documents the number of defects created during the completion of a particular task on the aircraft over time.

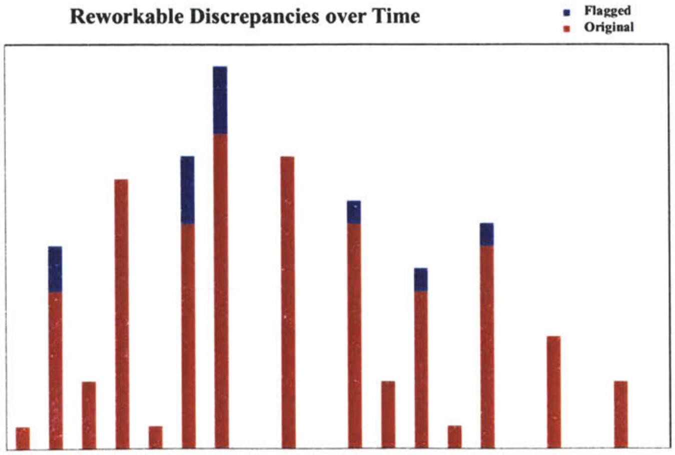


Figure 26: Example 1: High Variation

When the manager spoke with the supervisors who oversaw the completion of the task, he learned that it was typically alternated between two installers. Further, there had been one recent exception, when the same installer had performed the job twice in a row, exactly matching the pattern seen in the chart. Based on the data and these conversations, the production manager organized a meeting with the two installers and their supervisors to discuss the job and how to improve the performance of the second installer. During this meeting, it became apparent that the variability stemmed from the installers possessing a different understanding of a proper installation. With the data in front of them, this conversation was productive in sharing a best practice and removing an easily avoidable source of rework.

8.4.2 Comparing similar tasks

On another occasion, a line manager and supervisor were trying to work with an installer to improve the quality of his work. The installer in question had become upset, believing that his work was high in quality and that he was being unfairly targeted. At this point, the line manager selected the chart for this installer's installation and one for an identical installation on the opposite side of the aircraft, and presented the two to the installer. For convenience, these two charts have been superimposed as Figure 27.

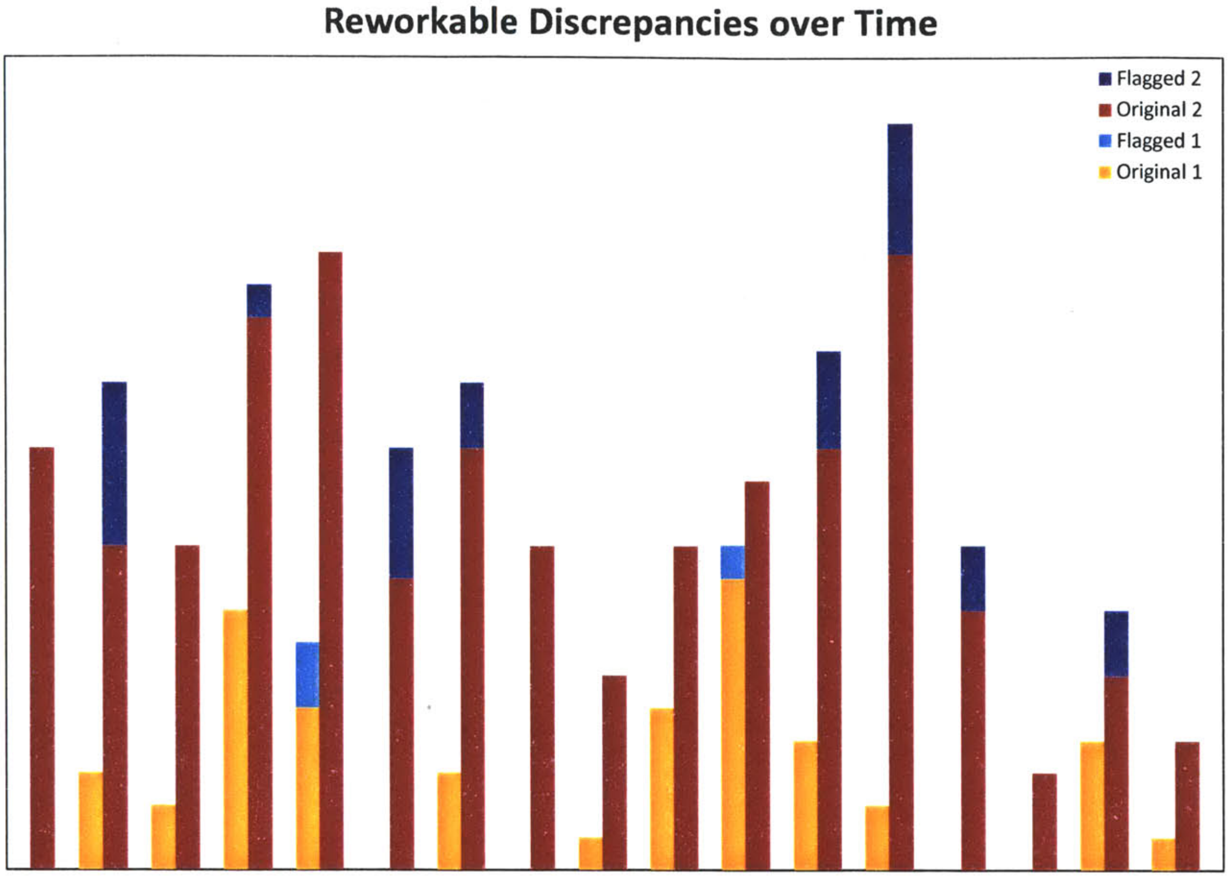


Figure 27: Example 2, Comparing Similar Tasks

As can be seen in the chart, not only did this installer's work result in a higher number of discrepancies it also resulted in a higher number of "flagged" discrepancies, resulting in additional work. Having the data enabled the line manager to bypass the defensive response from the installer who could instead focus on how to improve his performance. With the data taking a key role in the meeting, the installer and manager were able to develop a plan to improve performance, with the new metric providing data support to monitor progress and provide feedback.

8.4.3 Tasks with High Rework

As part of the beta testing phase, each of the line managers was provided with a complete run of the program, including the number of discrepancies being created on every task on their line, organized from worst to best. Some of the managers used this data and launched investigations into the poorest performing tasks, trying to understand the underlying causes of rework on these jobs.

8.5 Permanent Roll-Out

Prior to a roll-out program, the team tested the program extensively to identify and remove bugs. At the end of the author's internship, this beta testing was complete and the program was ready for roll-out to the industrial engineering group. It was determined that normal operation of the tool should include updating the reports independently for each line, with updates coming every time the line rolls. The only exception would be the report for the director, which would be run on a weekly basis, because it would require the combination of data across multiple lines.

Chapter 9: Discussion and Results

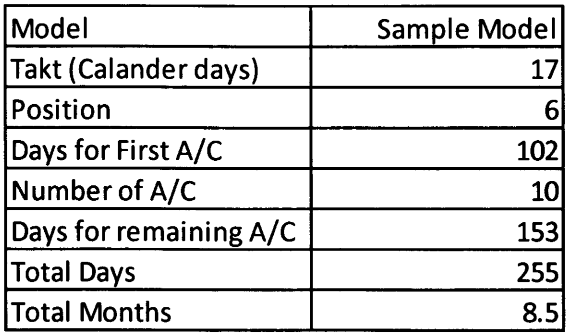
9.1 Convergence of Efforts

In chapters 5-8, a number of the different methods used to reduce rework were discussed. Fundamentally all of these efforts were intended to achieve the same goal, while using different means to pursue the problem. In this way, even if one (or all) of the target teams or the systemic approach failed, the project would still be successful in reducing the amount of rework. Additionally, there was the possibility that these efforts might combine to produce a greater effect than each individual project. In fact, most of the efforts succeeded although a few methods identified by the team were ultimately abandoned because they proved unsuccessful while others were put on hold to be pursued in the future. Section 9.4 summarizes the impact that all of these efforts have achieved to date.

Another commonality between all of the methods pursued by the target team was the focus put into the exact tactics used to achieve the results. As discussed in Chapter 2 and Chapter 4, completing a successful change effort would require the team to work within the culture of Raptor Aerospace, while at the same time trying to change the culture to have a sustained long term impact. Given the interplay between the different parts the project, there was a risk that pushing through one part, over other objections, might create resistance in other areas, resulting in some potentially effective rework reduction ideas being classified as a failure. In fact, one of the target teams chose to pursue a topic that had previously been the focus of two different initiatives. However, the project team hoped that in this case, the positive good will that had been created by the project to date would enable this effort to succeed where others had failed before, an assessment that is ongoing. The overall impact that the project has had on the culture of Raptor Aerospace will be discussed in section 9.6.

9.2 Time to Measure Results

In implementing corrective actions, the team observed that some produced immediate results, such as the type shown in section 6.4 following the implementation of corrective actions for wedged clamps. In these cases, the results of the change were obvious very quickly. On the other hand, if a corrective action had a slower implementation or did not produce a dramatic decrease in rework, the impact of this change might be disguised by normal variation in rework between aircraft. In order, to demonstrate if this type of corrective action was effective, the team would need to monitor multiple aircraft to determine if there was a statistically significant change. For example, Table 8 demonstrates a sample calculation for how long the team would need to wait if 10 data points were needed. In this sample, with a takt time of an average of 17 days and 6 positions on the line, it would take eight and a half months before 10 aircraft will pass through the entire final assembly line, and a small change will be confirmed as having an impact.



Model	Sample Model
Takt (Calander days)	17
Position	6
Days for First A/C	102
Number of A/C	10
Days for remaining A/C	153
Total Days	255
Total Months	8.5

Table 8: Time to Measure Impact

9.3 External Influences on Rework

One of the most difficult challenges in measuring the impact of a rework reduction project is isolating the efforts of the project from other external factors. For example, as the project progresses, efforts from the project team increase inspectors' and installers' awareness of the issue. This knowledge can have numerous effects on the amount of rework at Raptor. Installers may generate a reduced number of defects because of greater focus on the issue, while inspectors showing greater vigilance may reduce the inherent Type II error rate, giving the appearance of an increase in errors. While one might expect these two examples to have only a short term impact, they both make it more difficult to determine the real long term impact of changes being made. Additionally, while the team may be aware of some of the external factors which effect the rework rate, it is nearly impossible to isolate and identify every external factor. Furthermore, even when the team is aware of external factors, measuring the size of their impact is frequently impossible.

One example of the later situation occurred during the rework reduction project when Raptor underwent a major restructuring of its entire manufacturing organization, including the installation crews. One of the major repercussions of this restructuring is that numerous jobs that were previously being completed by a particular group of installers changed. In the past, when installers worked on new jobs, there was a spike in the amount of rework needed until the new worker had gained enough experience to match the performance of the previous installer. However, in the immediate aftermath of this change, the team observed no significant change in the amount of rework at Raptor.

9.4 Achieved Savings

At the request of Raptor Aerospace, the savings values that were achieved in each action taken by the project team will be lumped into a single category, and the overall value of the project will be discussed as well as ongoing projects and potential future directions. Additionally, many of the efforts described in this paper are still ongoing and/or are in the initial phase, and likely have not reached their full potential at the time of the writing of this thesis. The first initiatives for the project were implemented approximately three and a half months after the launch of the program. Over the course of the next three months, additional recommendations were implemented and began contributing to a reduction in the amount of rework at Raptor Aerospace. All told the efforts have had the following impact:

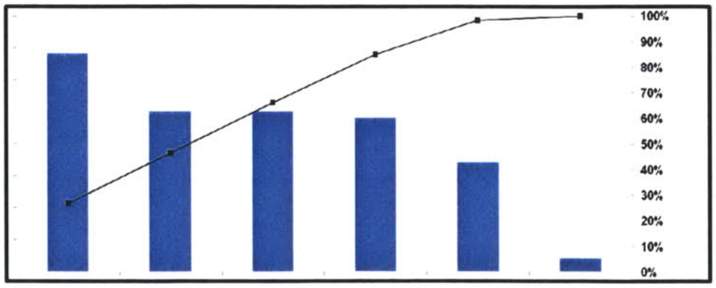
- Verified savings of over **$350,000** during the authors' time working with Raptor
- Verified annualized savings of over **$2,000,000**
- Additional unverified savings
- Savings of over 40 man-hours a week through automation of tasks
- Significant reduction in the total number of discrepancies per year

In addition to the savings that have already been achieved, efforts that were initiated by the project and are still ongoing have an expected savings of well over $10,000,000 based upon their current progress. Furthermore, other opportunities that have already been identified but not targeted have the potential to increase the total savings by another $15,000,000 to $20,000,000. Overall, the project is considered by Raptor to have been successful in achieving its objectives, realizing significant savings, freeing up large amounts of employee time through automation and influencing the culture of Raptor. However, the continued success of the rework reduction efforts will depend upon the continuing dedication of the project team to work on driving the projects and adapting the culture of Raptor to a more competitive future.

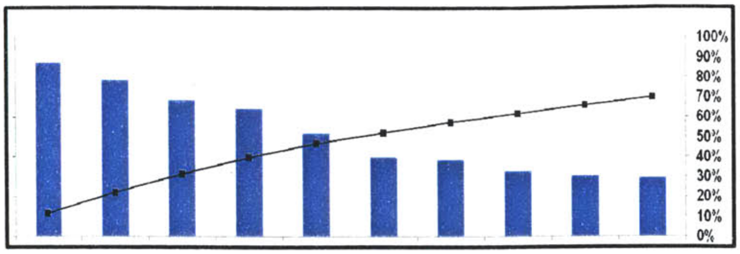
9.5 Implementation of Project Methods in Other Departments

Toward the end of the author's time at the site, news about the current successes of the project team began to disseminate throughout the company, and other departments approached the rework reduction team to determine if the approach used for rework reduction could be extended to their specific projects. To this end, the primary data analysis methods described in Chapter 3 can be adapted for any dataset provided that either the input data are adjusted to match the input fields or the database tools are adjusted to the new data fields (the more common approach). Typically, adjusting the tools to process a new dataset takes approximately 2-4 hours (assuming the update is done by someone with experience in VBA), with another hour of interpretation time to validate the results and pick the final categories for the cross pareto.

In one example, an engineer in another department at Raptor was asked for a better understanding of a dataset recording a request for information that is regularly posed to his group. At the time, this database was examined manually on a monthly basis and took approximately 24 hours per month to create a pareto of top issues. This analysis can be seen in Figure 28, which depicts this pareto (but only includes data points that fit within the identified key topics).



Working with the project team, the engineer ran the dataset through the Word Pareto to identify key terms which were examined and classified. Next, these terms were inserted into the cross pareto tool using methods similar to those described in Chapter 3.4. The analysis generated from this method identified several new opportunities that had not been discovered previously, and also provided far greater insight into each of the topics that was highlighted. These results are included in Figure 29, which depicts the top 10 issues that were identified and represent approximately 70% of the entire dataset. It is important to note that in this pareto, the relative impact of these ten issues to the total number of data points is clear, unlike the previous chart.



Once the initial analysis was complete, the program was handed over to the engineer to maintain its operation. In this role, he determined that the new method would need less than 2 hours a month to operate, representing a significant time savings (22 man-hours a month), while identifying more issues and providing greater insight into each of those topics. Ultimately, the time spent adapting the program to the new dataset was easily recovered in its first month of use by the group and has instigated the spread of the tools to numerous groups throughout the company.

9.6 Cultural Impact of Project to Date

In Section 2.3, the project team's goals for changes in both the short-term and long-term cultures were discussed. Further, chapter 4 addressed some of the approaches taken by the project team to successfully navigate through some of the hurdles present in Raptor's current environment. Overall, these efforts have been successful, as the project team continues to receive support and praise from the different departments alongside which it has worked. Beyond delivering a reduction in the amount of rework at Raptor Aerospace, the project team also believed that it has seen indications of changing behavior, which could signify improvements within the culture. In section 2.2.5, five aspects of Raptor's culture were identified which had a negative impact on improvement efforts at the company. To date, there are indications of improvements in all these areas including departmental prejudice, poor communication, finger-pointing, firefighting, and individual resistance to change.

In the first of these issues, departmental prejudice, the team has successfully used data to quantify several of the prejudices that existed. In doing so, it has enabled these departments to understand the true sources of these opinions and thus, to address these underlying causes. Furthermore, when prejudices were not justified by the data, it enabled different groups to discuss these issues in a more factual and less emotional way. For the second topic, poor communication, the team has heard anecdotal stories of improvement, such as the one following the wedge clamp specification change. Progress was also achieved in the third issue, finger-pointing, as the different departments successfully worked together on several projects within the rework reduction effort. Additionally, the team successfully motivated each of the departments to take responsibility for delivering results in different parts of the project, instead of trying to pass responsibility on to another group.

In the fourth topic, firefighting, the financial success that has been achieved to date, along with widespread accolades for the project, has created an increase in interest in preventing problems. Not only have other departments approached the project team to adopt similar efforts, but individuals within final assembly have approached the project team with hopes of leading future target teams. In the final issue, individual resistance to change, the team has found that it easier to get ideas and suggestions from other groups as time has progressed. The team attributes this change to the fact that the project has created visible improvements and remains supported and led by motivated individuals.

Overall, while there have been changes in the company culture, it has not yet reached the team's short-term goals. With this being the case, it is likely that this progress is still fragile, and unless vigilance is maintained could easily slip back into the situation that was present at the beginning of the project.

9.7 Ongoing Efforts

In the time since the author departed Raptor Aerospace, efforts to reduce rework have continued unabated. The teams that were launched (but not completed) during the internship continue to progress and a third round of teams have since been launched. Additionally, the strategy team is planning to create target teams at sister facilities in an attempt to expand the project across the entire company. Of particular note, a new production line in one of the sister facilities approached the strategy team about leading one of these teams (further evidence of cultural improvements). Once the systemic approach described in chapter 8 is completely launched, the team will begin work on the next systemic effort to reduce rework at Raptor Aerospace. Finally, the team will continue to look for new methods to reduce rework and create a lasting change at the company. Maintaining continuity within the project team is fundamental to its success. With the author of this thesis leaving the site, the project team is reduced to one member who has been present since the beginning of the project, making his contributions critical.

9.8 Assessment of Hypothesis

Overall, the author believes that it is still too early in the project to evaluate the hypothesis conclusively. However, the financial success that has been achieved to date, combined with the evidence of cultural improvements, is encouraging and represents an unambiguous first step toward creating a self-sustaining improvement effort at Raptor Aerospace. In particular, the fact that other groups at Raptor have taken the initiative to approach the project team about participating in or leading parts of the project represents a key milestone. If these groups learn this methodology, incorporate it into their culture, and are driven to develop new efforts in the future, the project will be deemed a complete success.

Furthermore, it would appear that the approaches taken at Raptor could be successfully implemented at other companies within and outside of the aerospace industry. However, in applying these methods, it is critical to incorporate the individual nuances of these companies and not just apply the methods blindly. This is one of the most important lessons from this project.

Works Cited

Achanga, P., Shehab, E., Roy, R., & Nelder, G. (2006). Critical success factors for lean implementation within SMEs. *Journal of Manufacturing Technology Management, 17*(4), 460.

Akbulut-Bailey, A., Motwani, J., & Smedley, E. M. (2012). When lean and six sigma converge: A case study of a successful implementation of lean six sigma at an aerospace company. *International Journal of Technology Management, 57*(1), 18-32.

Assum, T., Bjørnskau, T., Fosser, S., & Sagberg, F. (1999). Risk compensation—the case of road lighting. *Accident Analysis & Prevention, 31*(5), 545-553.

Axelsson, J. R. C. (2000). *Quality and ergonomics : Towards successful integration*. Linköping: Univ.

Barber, P., Graves, A., Hall, M., & Tomkins, C. (2000). Quality failure costs in civil engineering projects. *The International Journal of Quality & Reliability Management, 17*(4/5), 479-492.

Barrie, J. M. (James Matthew). (1991). *Peter pan: Peter pan and wendy* (Millennium Fulcrum ed.). Retreived from Project Gutenberg: http://www.gutenberg.org/ebooks/16

Bendell, T. (2006). A review and comparison of six sigma and the lean organisations. *The TQM Magazine, 18*(3), 255.

Bonavolonta, C., Valentino, M., Peluso, G., & Barone, A. (2007). Non destructive evaluation of advanced composite materials for aerospace application using HTS SQUIDs. *Applied Superconductivity, IEEE Transactions on, 17*(2), 772-775.

Browning, T. R., & Heath, R. D. (2009). Reconceptualizing the effects of lean on production costs with evidence from the F-22 program. *Journal of Operations Management, 27*(1), 23-44.

Burati, J.,James L., Farrington, J. J., & Ledbetter, W. B. (1992). Causes of quality deviations in design and construction. *Journal of Construction Engineering and Management, 118*(1), 34-49.

BURKE, R. J., DAVIS, R. D., KAMINSKY, F. C., & ROBERTS, A. E. P. (1995). THE EFFECT OF INSPECTOR ERRORS ON THE TRUE FRACTION NON-CONFORMING: AN INDUSTRIAL EXPERIMENT. *Quality Engineering, 7*(3), 543-550.

Campanella, J. (Ed.). (1999). *Principles of quality costs: Principles, implementation and use* (3rd ed.). Milwaukee, WI: American Society for Quality, Quality Cost Committee.

Clark, D. L., Howell, D. M., & Wilson, C. E. (2007). IMPROVING NAVAL SHIPBUILDING PROJECT EFFICIENCY THROUGH REWORK REDUCTION. (Masters, NAVAL POSTGRADUATE SCHOOL). *The Defense Technical Information Center,*

Cooper, K. G. (1993). The rework cycle: Benchmarks for the project manager (part 3). *Project Management Journal, 24*(1), 17.

Cooper, K. G. (1994). The $2,000 hour: How managers influence project performance through the rework cycle. *Project Management Journal, 25*(1), 11.

Crute, V., Ward, Y., Brown, S., & Graves, A. (2003). Implementing lean in aerospace—challenging the assumptions and understanding the challenges. *Technovation, 23*(12), 917-928.

De Angelis, G., Meo, M., Almond, D. P., Pickering, S. G., & Angioni, S. L. (2012). A new technique to detect defect size and depth in composite structures using digital shearography and unconstrained optimization. *NDT & E International, 45*(1), 91-96.

Dostaler, I. (2010). Avoiding rework in product design: Evidence from the aerospace industry.*27*(1), 5-26.

Drucker, P. F. (1988, Aug 2). Workers' hands bound by tradition. *Wall Street Journal (1923 - Current File),* pp. 20.

DRURY, C. G., & SHEEHAN, J. J. (1968). ERGONOMIC AND ECONOMIC FACTORS IN AN INDUSTRIAL INSPECTION TASK. *International Journal of Production Research, 7*(4), 333-341.

Drury, C. G., Spencer, F. W., & Schurman, D. L. (1997). Measuring human detection performance in aircraft visual inspection. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 41*(1), 304-308.

Fisher, M. L., & Ittner, C. D. (1999). The impact of product variety on automobile assembly operations: Empirical evidence and simulation analysis. *Management Science, 45*(6), 771.

Geswein, B. C., Sloan School of Management, Massachusetts Institute of Technology. Dept. of Mechanical Engineering, & Leaders for Global Operations Program. Thanks for using Barton,the MIT Libraries' catalog http://libraries.mit.edu/barton. (2011). Optimization of shake inspections. (M.B.A.)--Massachusetts Institute of Technology, Sloan School of Management; and, (S.M., Massachusetts Institute of Technology, Dept. of Mechanical Engineering; in conjunction with the Leaders for Global Operations Program at MIT)., 90.

Harrington, H. J. (1987). *Poor-quality cost.* New York; Milwaukee: M. Dekker ; ASQC Quality Press.

Hegazy, T., Said, M., & Kassab, M. (2011). Incorporating rework into construction schedule analysis. *Automation in Construction, 20*(8), 1051-1059.

Honeywell Aerospace. (2001). *Honeywell quality management operations.* Retrieved 02/20, 2012, from http://www.honeywell.com/sites/servlet/com.merx.npoint.servlets.DocumentServlet?docid=D1CAE016E-30AC-69DF-FA95-2410C2D2C01D

Itoh, M., Sakami, D., & Tanaka, K. (2007). Risk compensation due to human adaptation to automation for system safety. *Transactions of the Society of Instrument and Control Engineers, 43*(10), 926-34.

Iyengar, S. S., & Lepper, M. R. (2000). When choice is demotivating: Can one desire too much of a good thing? *Journal of Personality and Social Psychology, 79*(6), 995.

JARAIEDI, M., KOCHHAR, D., & JAISINGH, S. (1987). Multiple inspections to meet desired outgoing quality. *Journal of Quality Technology, 19*(1), 46-51.

Krishnan, S. K. (2006). Increasing the visibility of hidden failure costs. Measuring Business Excellence, 10(4), 77.

Lewis, M. A. (2000). Lean production and sustainable competitive advantage. *International Journal of Operations & Production Management, 20*(8), 959.

Lockheed Martin. *Lockheed martin quality assurance program.* Retrieved 02/20, 2012, from http://www.lockheedmartin.com/us/products/seaporte/quality-assurance-program.html

Love, P. E. D. (2002). Auditing the indirect consequences of rework in construction: A case based approach. *Managerial Auditing Journal, 17*(3), 138-146.

Love, P. E. D., & Edwards, D. J. (2004). Forensic project management: The underlying causes of rework in construction projects. *Civil Engineering and Environmental Systems, 21*(3), 207-228.

Love, P. E. D., Edwards, D. J., Watson, H., & Davis, P. (2010). Rework in civil infrastructure projects: Determination of cost predictors. *Journal of Construction Engineering & Management, 136*(3), 275-282.

Love, P. E. D., & Li, H. (2000). Quantifying the causes and costs of rework in construction. *Construction Management & Economics, 18*(4), 479-490.

M., B., & L., S. (2003). How to identify and remove the barriers for a successful lean implementation. *Journal of Ship Production, 19*(2), 116-120.

Maleyeff, J., Kaminsky, F., & Farris, J. (2003). Minimum cost 100% inspection systems with inspection error. *Quality Engineering, 15*(4), 557.

MEI, W., CASE, K. E., & SCHMIDT, J. W. (1975). Bias and imprecision in variables acceptance sampling effects and compensation. *International Journal of Production Research, 13*(4), 327-340.

Moselhi, O., Assem, I., & El-Rayes, K. (2005). Change orders impact on labor productivity. *Journal of Construction Engineering & Management, 131*(3), 354-359.

National Transportation Safety Board. (2011). *Accidents, fatalities, and rates, 1991 - 2010, 14 CFR 121, scheduled and nonscheduled service (airlines).* Retrieved March 7, 2012, from http://www.ntsb.gov/data/aviation_stats.html

Northrop Grumman. (2009). *Northrop grumman quality assurance.* Retrieved 02/20, 2012, from http://www.is.northropgrumman.com/products/epower/customer_care/asset/Quality_Assurance_BPS-029.pdf

Omachonu, V. K., Suthummanon, S., & Einspruch, N. G. (2004). The relationship between quality and quality cost for a manufacturing company. *The International Journal of Quality & Reliability Management, 21*(2/3), 277.

Osder, S. (1999). Practical view of redundancy management application and theory. *Journal of Guidance, Control, and Dynamics, 22*(1), 12-21.

Owens, B. D., Leveson, N. G., & Hoffman, J. A. (2011). Procedure rework: A dynamic process with implications for the ?rework cycle? and ?disaster dynamics? *System Dynamics Review, 27*(3), 244-269.

Peikert, A., Thoma, J., & Brown, S. (1998). A rapid modeling technique for measurable improvements in factory performance. *Proceedings of the 30th Conference on Winter Simulation,* Washington, D.C., United States. pp. 1011-1016.

Peltzman, S. (1975). The effects of automobile safety regulation. *Journal of Political Economy, 83*(4), 677.

Phillips, R. O., Fyhri, A., & Sagberg, F. (2011). Risk compensation and bicycle helmets. *Risk Analysis, 31*(8), 1187-1195.

Robbins, H., & Finley, M.,. (1996). *Why change doesn't work : Why initiatives go wrong and how to try again--and succeed.* Princeton, N.J.: Peterson's.

Sim, K. L., & Rogers, J. W. (2009). Implementing lean production systems: Barriers to change. *Management Research News, 32*(1), 37.

Tsai, W. (1998). Quality cost measurement under activity-based costing. *The International Journal of Quality & Reliability Management, 15*(7), 719.

United States Department of Labor, Bureau of Labor Statistics. (2012). CPI inflation calculator. Retrieved April 15, 2012, from http://www.bls.gov.libproxy.mit.edu/data/inflation_calculator.htm

United Technology Corporation. (2012). *United technology corporation quality assurance.* Retrieved 02/20, 2012, from http://utc.com/About+UTC/Quality

Vassilakis, E., & Besseris, G. (2009). An application of TQM tools at a maintenance division of a large aerospace company. *Journal of Quality in Maintenance Engineering, 15*(1), 31.

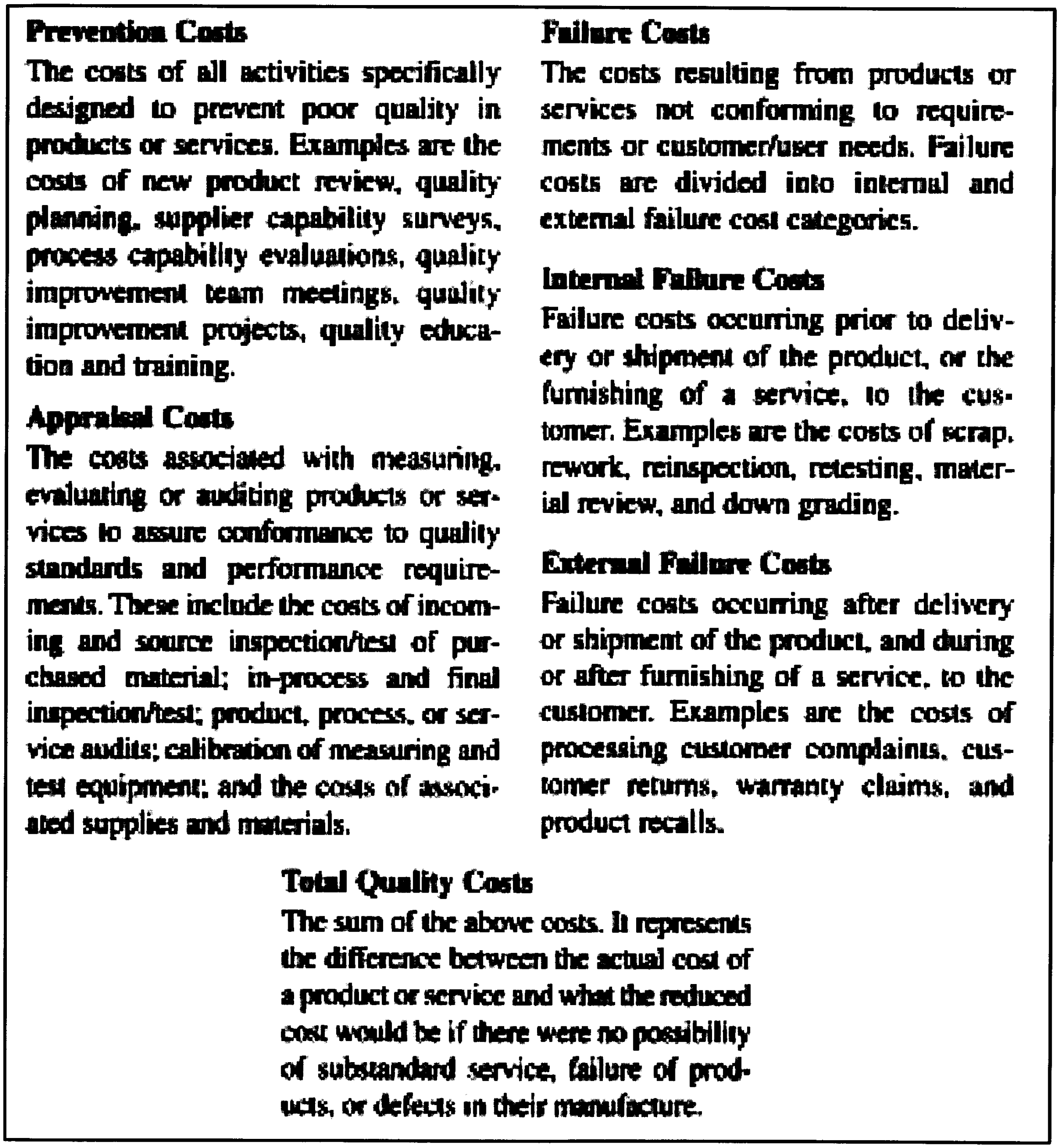
Vaughen, B. K., & Muschara, T. (2011). A case study?combining incident investigation approaches to identify system-related root causes. *Process Safety Progress, 30*(4), 372-376.

VELOCCI JR., . (1998). Cost of quality an industry challenge. *Aviation Week and Space Technology, 149*(20), 58.

Womack, J. P., Jones, D. T., & Roos, D. (2007). *The machine that changed the world : The story of lean production -- toyota's secret weapon in the global car wars that is revolutionizing world industry* (1 trade pbk ed.). New York: Free press.

Appendix A

This appendix contains figures that were too large to be included in the body of the thesis. For particularly important figures, a smaller version was included to provide the general appearance of the object in question.



Prevention Costs

The costs of all activities specifically designed to prevent poor quality in products or services. Examples are the costs of new product review, quality planning, supplier capability surveys, process capability evaluations, quality improvement team meetings, quality improvement projects, quality education and training.

Appraisal Costs

The costs associated with measuring, evaluating or auditing products or services to assure conformance to quality standards and performance requirements. These include the costs of incoming and source inspection/test of purchased material; in-process and final inspection/test; product, process, or service audits; calibration of measuring and test equipment; and the costs of associated supplies and materials.

Failure Costs

The costs resulting from products or services not conforming to requirements or customer/user needs. Failure costs are divided into internal and external failure cost categories.

Internal Failure Costs

Failure costs occurring prior to delivery or shipment of the product, or the furnishing of a service, to the customer. Examples are the costs of scrap, rework, reinspection, retesting, material review, and down grading.

External Failure Costs

Failure costs occurring after delivery or shipment of the product, and during or after furnishing of a service, to the customer. Examples are the costs of processing customer complaints, customer returns, warranty claims, and product recalls.

Total Quality Costs

The sum of the above costs. It represents the difference between the actual cost of a product or service and what the reduced cost would be if there were no possibility of substandard service, failure of products, or defects in their manufacture.

Figure 30: Quality Costs (Campanella, 1999)

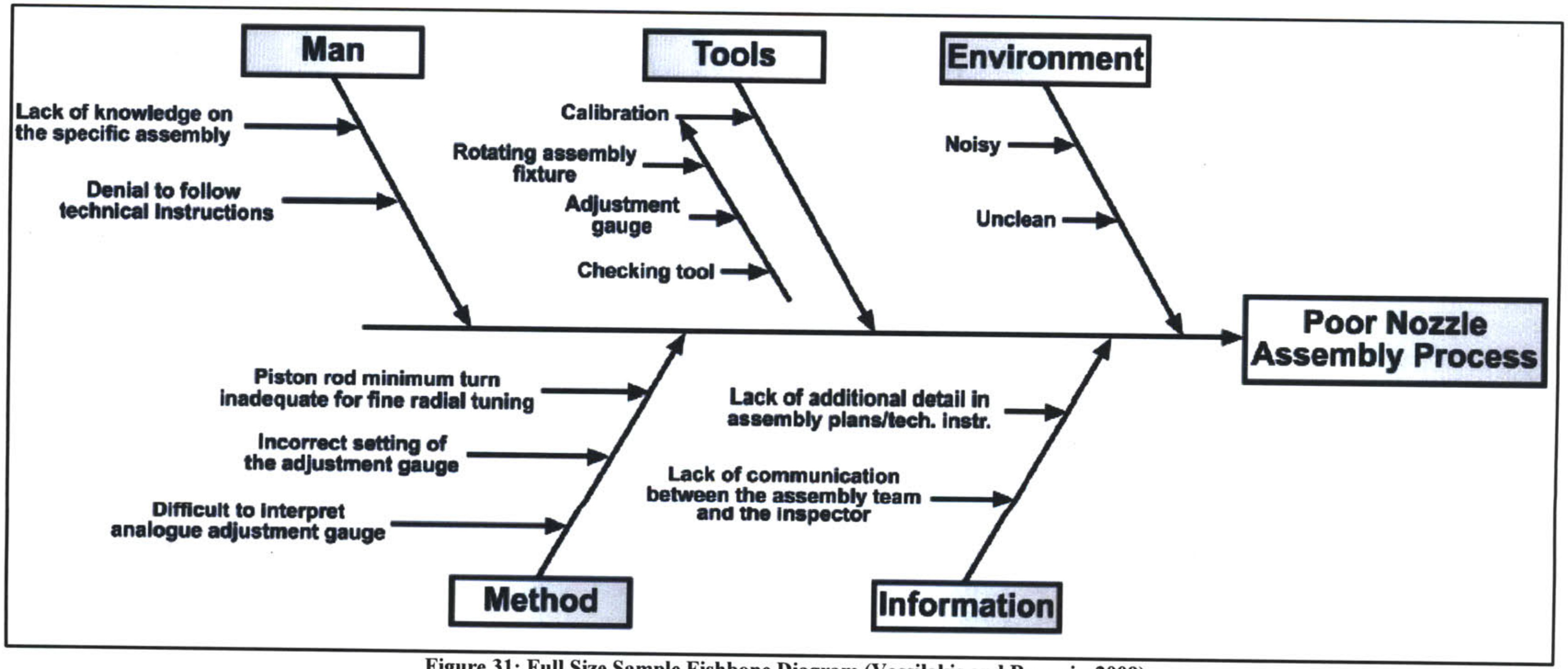
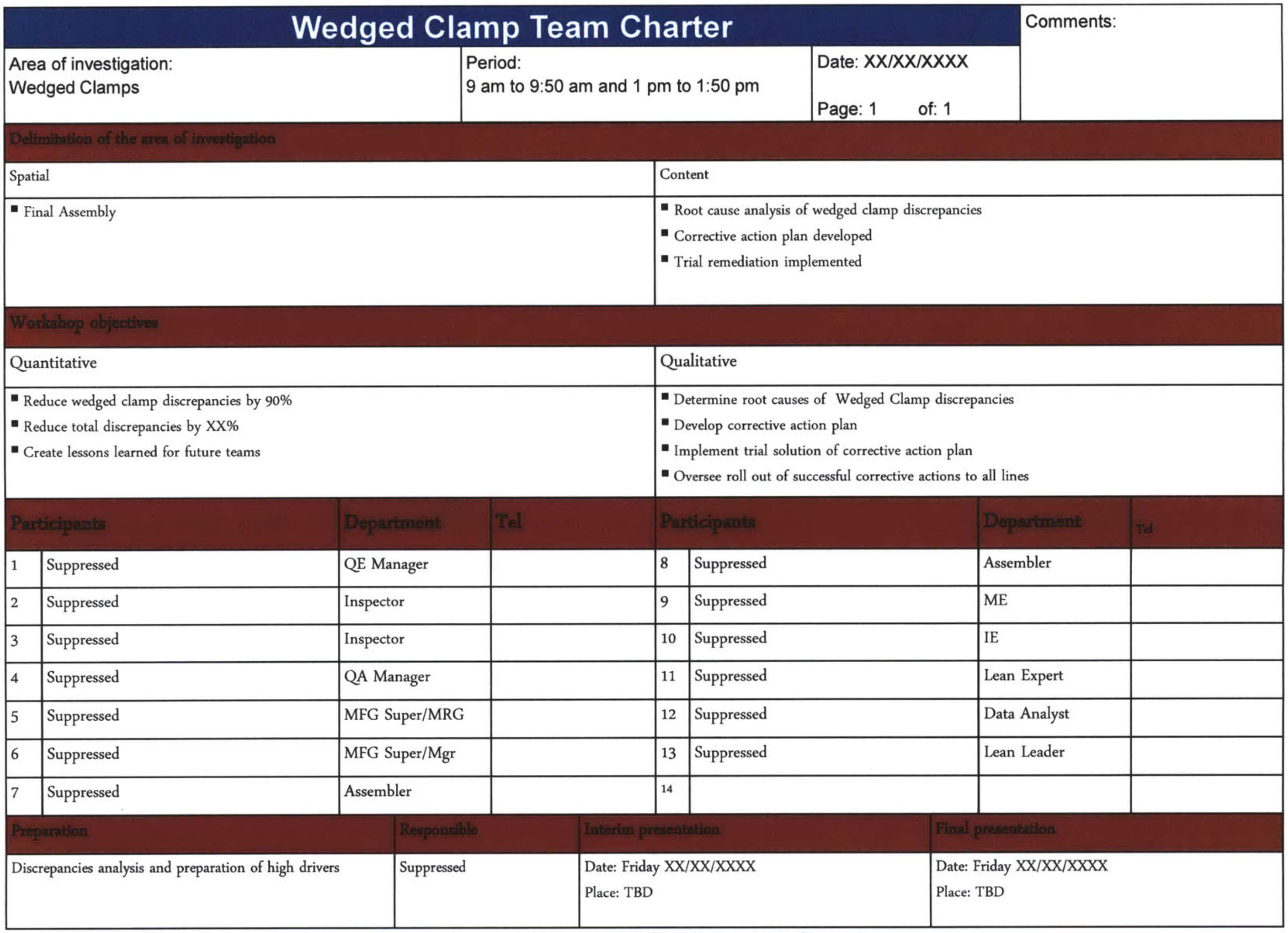

Man
Tools
Environment
Lack of knowledge on the specific assembly
Denial to follow technical instructions
Calibration
Rotating assembly fixture
Adjustment gauge
Checking tool
Noisy
Unclean
Poor Nozzle Assembly Process
Piston rod minimum turn inadequate for fine radial tuning
Incorrect setting of the adjustment gauge
Difficult to interpret analogue adjustment gauge
Lack of additional detail in assembly plans/tech. instr.
Lack of communication between the assembly team and the inspector
Method
Information

Figure 31: Full Size Sample Fishbone Diagram (Vassilakis and Besseris, 2009)



Wedged Clamp Team Charter

Area of investigation: Wedged Clamps	Period: 9 am to 9:50 am and 1 pm to 1:50 pm	Date: XX/XX/XXXX Page: 1 of: 1	Comments:

Delimitation of the area of investigation

Spatial	Content
■ Final Assembly	■ Root cause analysis of wedged clamp discrepancies ■ Corrective action plan developed ■ Trial remediation implemented

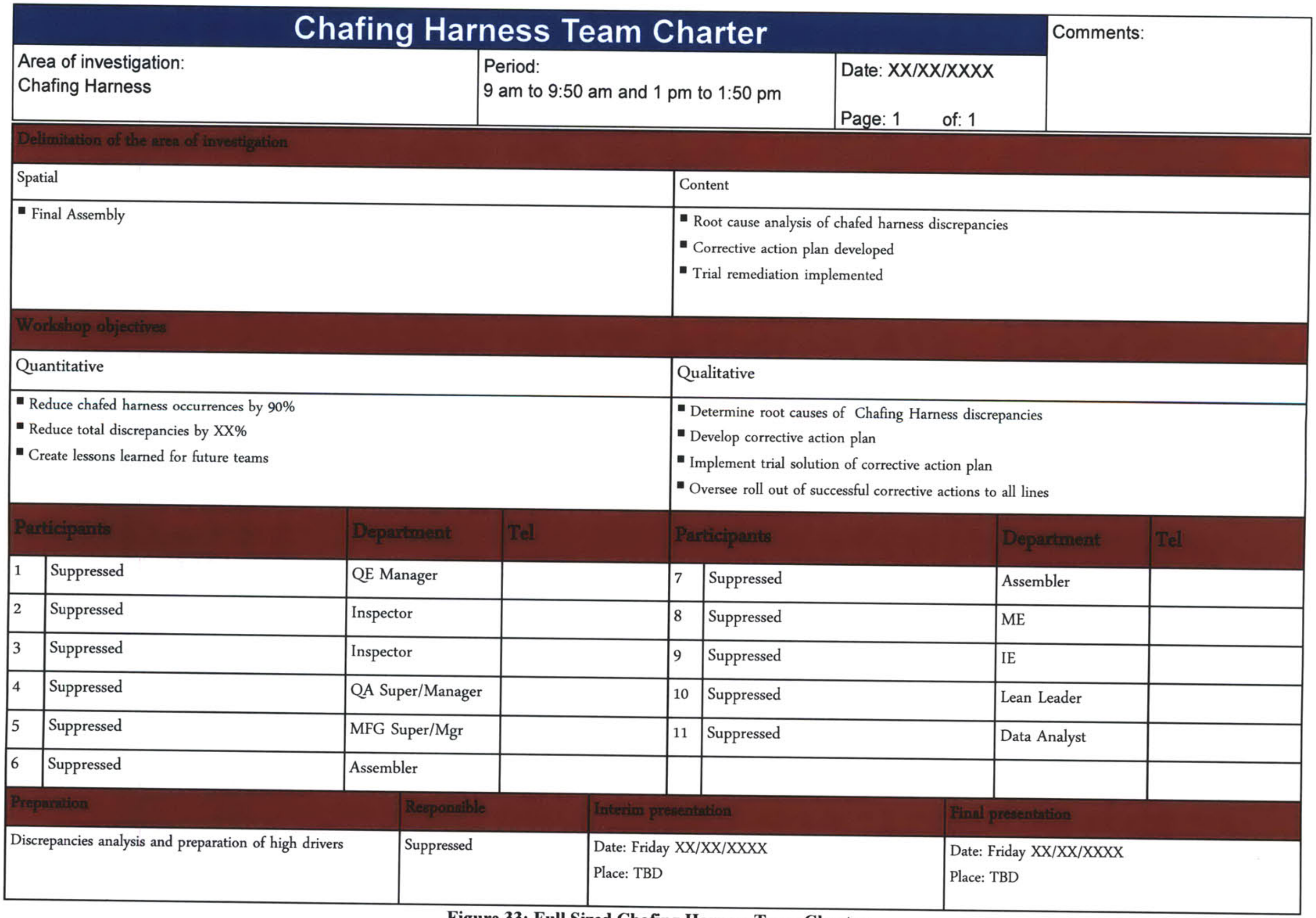
Workshop objectives

Quantitative	Qualitative
■ Reduce wedged clamp discrepancies by 90% ■ Reduce total discrepancies by XX% ■ Create lessons learned for future teams	■ Determine root causes of Wedged Clamp discrepancies ■ Develop corrective action plan ■ Implement trial solution of corrective action plan ■ Oversee roll out of successful corrective actions to all lines

Participants		Department	Tel	Participants		Department	Tel
1	Suppressed	QE Manager		8	Suppressed	Assembler	
2	Suppressed	Inspector		9	Suppressed	ME	
3	Suppressed	Inspector		10	Suppressed	IE	
4	Suppressed	QA Manager		11	Suppressed	Lean Expert	
5	Suppressed	MFG Super/MRG		12	Suppressed	Data Analyst	
6	Suppressed	MFG Super/Mgr		13	Suppressed	Lean Leader	
7	Suppressed	Assembler		14			

Preparation	Responsible	Interim presentation	Final presentation
Discrepancies analysis and preparation of high drivers	Suppressed	Date: Friday XX/XX/XXXX Place: TBD	Date: Friday XX/XX/XXXX Place: TBD

Figure 32: Full Sized Wedged Clamp Team Charter



Chafing Harness Team Charter

Area of investigation: Chafing Harness	Period: 9 am to 9:50 am and 1 pm to 1:50 pm	Date: XX/XX/XXXX Page: 1 of: 1	Comments:

Delimitation of the area of investigation	
Spatial	Content
■ Final Assembly	■ Root cause analysis of chafed harness discrepancies ■ Corrective action plan developed ■ Trial remediation implemented

Workshop objectives	
Quantitative	Qualitative
■ Reduce chafed harness occurrences by 90% ■ Reduce total discrepancies by XX% ■ Create lessons learned for future teams	■ Determine root causes of Chafing Harness discrepancies ■ Develop corrective action plan ■ Implement trial solution of corrective action plan ■ Oversee roll out of successful corrective actions to all lines

Participants		Department	Tel	Participants		Department	Tel
1	Suppressed	QE Manager		7	Suppressed	Assembler	
2	Suppressed	Inspector		8	Suppressed	ME	
3	Suppressed	Inspector		9	Suppressed	IE	
4	Suppressed	QA Super/Manager		10	Suppressed	Lean Leader	
5	Suppressed	MFG Super/Mgr		11	Suppressed	Data Analyst	
6	Suppressed	Assembler					

Preparation	Responsible	Interim presentation	Final presentation
Discrepancies analysis and preparation of high drivers	Suppressed	Date: Friday XX/XX/XXXX Place: TBD	Date: Friday XX/XX/XXXX Place: TBD

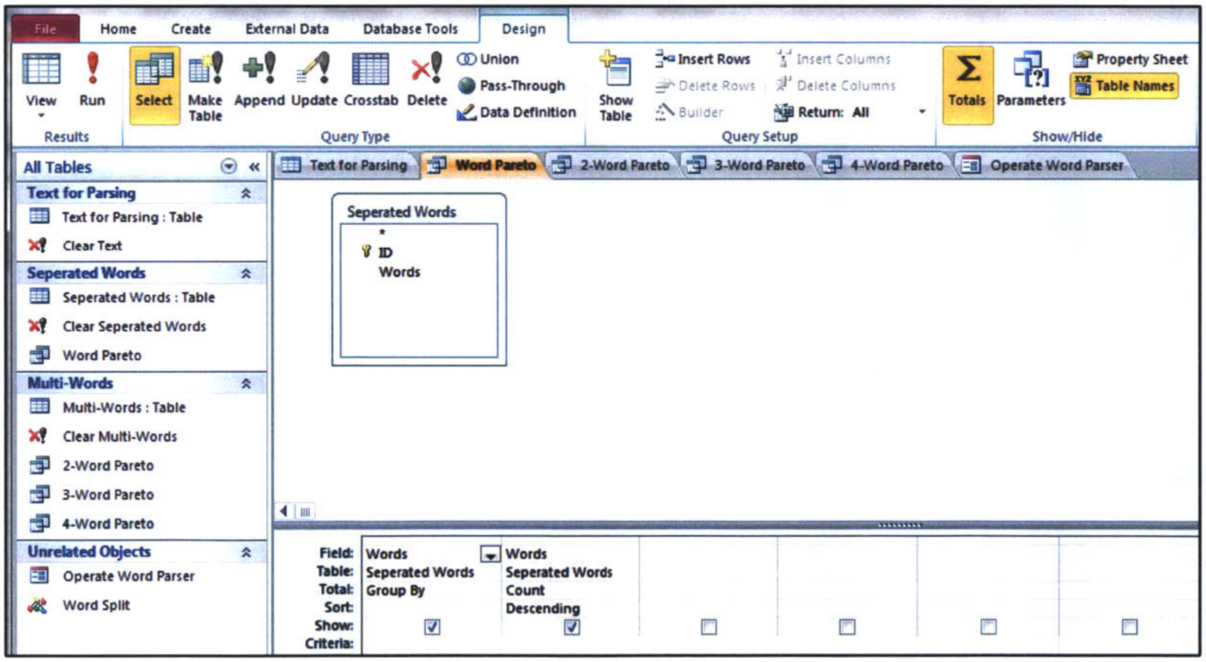
Figure 33: Full Sized Chafing Harness Team Charter

Appendix B

B.1 Word Parser Summary

The word parser was developed in Microsoft Access utilizing Visual Basic for Applications (VBA) to provide most of the program's functionality. The parser will sort through millions of lines of text, creating a word pareto identifying the most common words in the record set. Additionally, the program will create a pareto for two-word, three-word and four-word combinations. Once the data has been uploaded, the program is operated with the press of a single button.

The word parser uses VBA code to separate the text so that each word (and multi-word combination) is stored in a unique record in a separate table. Once the data has been sorted, a summation select query creates the different pareto's. Figure 34 shows the design of one of these queries and Section B.2 contains the code that is used by this program.



Field:	Words	Words				
Table:	Seperated Words	Seperated Words				
Total:	Group By	Count				
Sort:		Descending				
Show:	☑	☑	☐	☐	☐	☐
Criteria:						

Figure 34: Word Pareto Interface

B.2 Word Parser VBA Code

The code for the Word Parser is contained in the following three functions:

- Word_split: first part of the word parser, separates each word into its own cell
- Word_Combos: Creates cells with all two, three and four word combination
- Automatic_Process: Automates all of the word parser functions

```
Sub word_split()

'****************************************************************************************************
'****************************************************************************************************
'*******                                                                                    *********
'*******    This function manages the first part of the word parser, taking the data        *********
'*******    from the original source and separating out each word into its own record       *********
'*******    which can be used for the counting functions                                    *********
'*******                                                                                    *********
'****************************************************************************************************
'****************************************************************************************************
'*******  Created by Jeremy Lieberman   *******  December 9, 2011   *******   V1.0   ***************
'****************************************************************************************************

'******************************     Variable Declaration      **************************************

Dim dbtemp As DAO.Database      'stores the database for this function
Dim rstdata As DAO.Recordset    'stores the "Text for Parsing" table which contains the raw data
Dim rstout As DAO.Recordset     'stores the "Seperated Words" table which contains the parsed results

Dim start As Integer            'records the position in the string where the word starts
Dim letter As String            'stores each word
Dim CurrentLetter As Integer    'records the letter currently being analyzed
Dim LastLetter As Integer       'stores the length of each

'******************************     Primary Code       *******************************************

Set dbtemp = CurrentDb
Set rstdata = dbtemp.OpenRecordset("Text for Parsing", dbOpenTable)
Set rstout = dbtemp.OpenRecordset("Seperated Words", dbOpenTable)

rstdata.MoveFirst   'start at the first record

DoCmd.OpenQuery ("Clear Seperated Words")  'empties the seperated words table before starting

Do While rstdata.EOF = False
    start = 1
    CurrentLetter = 1

    'the following piece of code determines the end of each record before the file moves to the next
    If Nz(rstdata![Remarks]) <> "" Then
        LastLetter = Len(rstdata![Remarks])
    Else
        LastLetter = 0  'in the case that the particular record is empty
    End If

    Do While CurrentLetter <= LastLetter    'process each record until the last letter of the record
        letter = Mid(rstdata![Remarks], CurrentLetter, 1)             'pulls the next letter
        If letter = " " Or letter = Chr(13) Or letter = Chr(10) Then   'symbols that mark end of word

            If CurrentLetter - start > 1 Then   'checks if a word is selected and not a blank space
                With rstout
                    .AddNew
                    ![Words] = Mid(rstdata![Remarks], start, (CurrentLetter - start)) 'outputs new word
                    .Update 'saves the new word as a new record
                End With
            End If
            start = CurrentLetter + 1   'sets start to the next letter after this word
        End If
        CurrentLetter = CurrentLetter + 1
    Loop
        With rstout
            .AddNew
            ![Words] = Mid(rstdata![Remarks], start, (CurrentLetter - start))
                                                              'adds the last word from each record
            .Update
        End With
    rstdata.MoveNext    'moves to the next record
Loop

End Sub
```

```
Sub word_combos()
'****************************************************************************************************
'****************************************************************************************************
'*******                                                                                    *********
'*******    This function manages the second part of the word parser, taking the separated *********
'*******    words and creating a table with 2, 3 and 4 word combinations for the multi-    *********
'*******    word counting functions                                                         *********
'*******                                                                                    *********
'****************************************************************************************************
'****************************************************************************************************
'******* Created by Jeremy Lieberman   *******  December 9, 2011   *******   V1.0   ***************
'****************************************************************************************************

'******************************     Variable Declaration      ***************************************

Dim dbtemp As DAO.Database      'stores the database for this function
Dim rstdata As DAO.Recordset    'stores the separated word table which contains the raw data
Dim rstout As DAO.Recordset     'stores the multi-word table where the output will be stored

'the next 4 variables store each word
Dim firstword As String, secondword As String, thirdword As String, forthword As String

'******************************     Initial Variable Set      ***************************************
Set dbtemp = CurrentDb
Set rstdata = dbtemp.OpenRecordset("Seperated Words", dbOpenTable)
Set rstout = dbtemp.OpenRecordset("Multi-Words", dbOpenTable)

'starts the variables blank
firstword = ""
secondword = ""
thirdword = ""
forthword = ""

DoCmd.OpenQuery ("Clear Multi-Words")  'clears the multi-word table for data entry


'******************************     Primary Code         *******************************************

rstdata.MoveFirst

Do While rstdata.EOF = False

    If firstword <> "" And secondword <> "" Then        'waits until the first two words are filled
        With rstout
            .AddNew
            ![2-words] = firstword + " " + secondword   'add two word combo

            If thirdword <> "" Then                     'wait until the first three words are full
                ![3-words] = firstword + " " + secondword + " " + thirdword 'add three word combo

                If forthword <> "" Then                 'wait until all four words are full
                    ![4-words] = firstword + " " + secondword + " " + thirdword + " " + forthword
                                                                'add four word combo
                End If
            End If
            .Update     'add new record
        End With
    End If

    'the next section of code moves the selected words to the next set
    firstword = secondword
    secondword = thirdword
    thirdword = forthword
    If Nz(rstdata![Words], "xxxxxxxx") = "xxxxxxxx" Then
        forthword = ""
    Else
    forthword = rstdata![Words]
    End If
    rstdata.MoveNext
Loop

'after the loop is complete this will add the final couple of words of the file into the record
If secondword <> "" And thirdword <> "" Then
    With rstout
        .AddNew
        ![2-words] = secondword + " " + thirdword
        If forthword <> "" Then
        ![3-words] = secondword + " " + thirdword + " " + forthword
        End If
        .Update
    End With
End If

If thirdword <> "" And forthword <> "" Then
    With rstout
        .AddNew
        ![2-words] = thirdword + " " + forthword
        .Update
    End With
End If

End Sub
```

```
Sub Automatic_Process()
'***************************************************************************************************
'***************************************************************************************************
'*******                                                                                   *********
'*******    This function automates the entire word parser by calling the other two VBA    *********
'*******    functions and processing all of the select queries.  When it is complete the   *********
'*******    user is presented with all of the results.                                     *********
'*******                                                                                   *********
'***************************************************************************************************
'***************************************************************************************************
'*******  Created by Jeremy Lieberman   *******  December 9, 2011  *******   V1.0   ***************
'***************************************************************************************************

DoCmd.SetWarnings False         'This stops Access from asking for permission for each line of code

Application.SysCmd acSysCmdInitMeter, "Part 1: Sorting Data", 1000  'display progress bar

Application.SysCmd acSysCmdUpdateMeter, 0   'sets task bar

Call word_split                                      'run sub to create single word table
Application.SysCmd acSysCmdUpdateMeter, 100

Call word_combos                                     'run sub to create 2,3, and 4 word table
Application.SysCmd acSysCmdUpdateMeter, 300

DoCmd.OpenQuery ("Word Pareto")                      'process word pareto
Application.SysCmd acSysCmdUpdateMeter, 400

DoCmd.OpenQuery ("2-Word Pareto")                    'process 2 word pareto
Application.SysCmd acSysCmdUpdateMeter, 600

DoCmd.OpenQuery ("3-Word Pareto")                    'process 3 word pareto
Application.SysCmd acSysCmdUpdateMeter, 800

DoCmd.OpenQuery ("4-Word Pareto")                    'process 4 word pareto
Application.SysCmd acSysCmdUpdateMeter, 1000

DoCmd.SetWarnings True 'This stops Access from asking for permission for each line of code
Application.SysCmd acSysCmdRemoveMeter 'removes progress bar

End Sub
```

Appendix C

C.1 Cross Pareto Automation Tool Summary

The Cross Pareto Tool was developed in Microsoft Access utilizing Visual Basic for Applications (VBA) to provide additional functionality. The tool creates a comprehensive pivot of the data based upon which records contain key words. This program is intended to work in conjunction with the Word Parser Tool discussed in Appendix B to determine the key words to be used although this is not a requirement. Once the user has determined which terms are going to be prioritized, the Cross Pareto Tool enables the analysis of the dataset to create a heat map of these topics like the one shown in Chapter 3.

The tools function is split between Microsoft Access queries and VBA to provide automation and certain functions which cannot be accomplished inherently by Access. The Access queries include select queries to identify records containing key words, append queries to add marks to the records that contain key words, and append queries to create the completed cross pareto. In addition, the format of the tool enables additional analysis to be easily added into the basic interface as frequently the team used this tool to perform follow-up work. Due to the proprietary nature of the dataset (in particular the record's column labels), most of the VBA features cannot be included in this thesis. Likewise, a copy of the interface cannot be included. Chapter C.2 does provide a copy of the automation code and one other function to provide an indication of the workings of this tool.

C.2 Cross Pareto Automation Code

C.2.1 Automation Code

The Automation Code is split into the following seven functions:

- Autocode: Calls the other automation functions in the proper sequence
- Selectrawdata: This function loads the raw data into the Access database
- Updaterawroll: This function loads the roll data into the Access database

- Makerolldates: This function updates the roll date tables for each line
- Part_and_defect: This function processes each of the key terms
- Makeupdatedtable: This function creates the updated data table which is the basis for all of the data analysis
- Defectvspartpivot: This function creates the finished cross pareto

```
Sub Autocode()
'************************************************************************************************
'************************************************************************************************
'*******                                                                                *********
'*******   This is the primary function for controlling the generation of the defect   *********
'*******   vs part cross pivot.  It also adds roll data for the aircraft in order to    *********
'*******   enable functionality for the user who knows access query functions           *********
'*******                                                                                *********
'************************************************************************************************
'************************************************************************************************
'******* Created by Jeremy Lieberman   ******* December 9, 2011   *******  V1.0   ***************
'************************************************************************************************

'******************************     Variable Declaration      *************************************

Dim Proceed As String                  'determines if the user wants to proceed

'******************************     Primary Code        ******************************************

'The next line of code selects the location where the exporting files will be saved, although currently
not in use

Proceed = MsgBox("Do you want to conduct the analysis  (this action cannot be undone, takes ~ 20
minutes)", vbYesNo)

If (Proceed = vbYes) Then

    DoCmd.SetWarnings False            'This stops Access from asking for permission for each line of code
    Call selectrawdata                 'Updates the raw data table for processing
    Call updaterawroll                 'updates the raw roll data table for processing
    Call makerolldates                 'use the raw data table to insert roll dates
    Call part_and_defect               'updates the parts and defect queries
    Call makeupdatedtable              'creates the updated data table and adds part and defect identifiers
    Call defectvspartpivot             'finishes the analysis by creating the defect vs part pivot

    DoCmd.SetWarnings True 'This stops Access from asking for permission for each line of code
End If

End Sub
```

```
Sub selectrawdata()
```

```
'*****************************************************************************************************
'*****************************************************************************************************
'*******                                                                                    *********
'*******    This function updates the raw data table which will store the raw data that     *********
'*******    will be used in the part vs defect pivot table.                                 *********
'*******                                                                                    *********
'*****************************************************************************************************
'*****************************************************************************************************
'*******  Created by Jeremy Lieberman  *******  December 9, 2011  *******  V1.0  ***************
'*****************************************************************************************************

'******************************    Variable Declaration    ************************************

Dim strSQL As String            'stores instructions to create the delete query for the raw data
Dim db As Database              'stores this database for processing
Dim queryDef As queryDef        'used in the creation of the temporary delete query

Dim filename As String          'stores the file where the raw data to be imported is held

'******************************    Primary Code           ****************************************

'*****************************************************************************************************
'the next part of the code clears the raw_data table and deletes the query so it is not
'run by accident from the normal view
strSQL = "DELETE Raw_Data.* FROM Raw_Data; "

Set db = CurrentDb
Set qryDef = db.CreateQueryDef("temp", strSQL)
DoCmd.OpenQuery "temp"
db.QueryDefs.Delete "temp"

'*****************************************************************************************************
'current location of the raw data . . . to be updated to a selection protocol
filename = "\\pathway\raw data.xlsx"

DoCmd.TransferSpreadsheet acImport, , "Raw_Data", filename, True, "Raw_Data!"
db.TableDefs.Delete "Raw_Data$_ImportErrors"    'removes the record of errors

End Sub
```

```
Sub updaterawroll()
'*****************************************************************************************************
'*****************************************************************************************************
'*******                                                                                    *********
'*******    This function updates the raw roll data table to represent the current data     *********
'*******                                                                                    *********
'*****************************************************************************************************
'*****************************************************************************************************
'*******  Created by Jeremy Lieberman  *******  December 9, 2011  *******  V1.0  ***************
'*****************************************************************************************************

    Dim rolldatapath As String ' stores the location of the raw roll data from the share drive

    DoCmd.OpenQuery "Clear Raw Roll Data"

    'stores key historic data
    rolldatapath = "\\pathway\AC Current Position.xls"
    DoCmd.TransferSpreadsheet acImport, , "Raw_Roll_Data", rolldatapath, True,
                                          "Aircraft_Current_Position!"

    'stores recent data
    rolldatapath = "\\pathway2\AC Current Position.xls"
    DoCmd.TransferSpreadsheet acImport, , "Raw_Roll_Data", rolldatapath, True,
                                          "Aircraft_Current_Position!"

End Sub
```

```
Sub makerolldates()
'**************************************************************************************************
'**************************************************************************************************
'*******                                                                                 *********
'*******    This function updates the roll dates table which stores information about    *********
'*******    when each of the lines being considered rolls                                *********
'*******                                                                                 *********
'**************************************************************************************************
'**************************************************************************************************
'******* Created by Jeremy Lieberman   ******* December 9, 2011   *******   V1.0   ***************
'**************************************************************************************************

'the next section of code updates the move dates queries to organize the data in the proper form
DoCmd.OpenQuery ("Clear Roll Dates")
DoCmd.OpenQuery ("XXXXXX Move Dates")
DoCmd.Close acQuery, "XXXXXX Move Dates"
DoCmd.OpenQuery ("YYYYYY Move Dates")
DoCmd.Close acQuery, "YYYYYY Move Dates"
DoCmd.OpenQuery ("ZZZZZZ Move Dates")
DoCmd.Close acQuery, "ZZZZZZ Move Dates"
DoCmd.OpenQuery ("WWWWWW Move Dates")
DoCmd.Close acQuery, "WWWWWW Move Dates"
DoCmd.OpenQuery ("VVVVVV Move Dates")
DoCmd.Close acQuery, "VVVVVV Move Dates"
DoCmd.OpenQuery ("UUUUUU Move Dates")
DoCmd.Close acQuery, "UUUUUU Move Dates"

'the last section runs the add queries to put the new data into the roll data tables
DoCmd.OpenQuery ("Add XXXXXX")
DoCmd.OpenQuery ("Add YYYYYY")
DoCmd.OpenQuery ("Add ZZZZZZ")
DoCmd.OpenQuery ("Add WWWWWW")
DoCmd.OpenQuery ("Add VVVVVV")
DoCmd.OpenQuery ("Add UUUUUU")

End Sub
```

```
Sub part_and_defect()
```

```
'****************************************************************************************************
'****************************************************************************************************
'*******                                                                                    *********
'*******    This function runs all of the part and defect separators, updating the data     *********
'*******    stored in these fields                                                          *********
'*******                                                                                    *********
'****************************************************************************************************
'****************************************************************************************************
'*******  Created by Jeremy Lieberman   *******  December 9, 2011   *******   V1.0   ***************
'****************************************************************************************************

    Application.SysCmd acSysCmdInitMeter, "Part 1: Sorting Data", 1000  'display progress bar

    'runs the part and defect queries to update the data stored in these queries to ensure accuracy
    DoCmd.OpenQuery "Part1"
    DoCmd.Close acQuery, "Part1"
    DoCmd.OpenQuery "Part2"
    DoCmd.Close acQuery, "Part2"
    DoCmd.OpenQuery "Part3"
    DoCmd.Close acQuery, "Part3"
    DoCmd.OpenQuery "Part4"
    DoCmd.Close acQuery, "Part4"
    DoCmd.OpenQuery "Defect1"
    DoCmd.Close acQuery, "Defect1"
    Application.SysCmd acSysCmdUpdateMeter, 100
    DoCmd.OpenQuery "Part5"
    DoCmd.Close acQuery, "Part5"
    DoCmd.OpenQuery "Part6"
    DoCmd.Close acQuery, "Part6"
    DoCmd.OpenQuery "Part7"
    DoCmd.Close acQuery, "Part7"
    DoCmd.OpenQuery "Defect2"
    Application.SysCmd acSysCmdUpdateMeter, 200
    DoCmd.Close acQuery, "Defect2"
    DoCmd.OpenQuery "Part8"
    DoCmd.Close acQuery, "Part8"
    DoCmd.OpenQuery "Defect3"
    DoCmd.Close acQuery, "Defect3"
    Application.SysCmd acSysCmdUpdateMeter, 300
    DoCmd.OpenQuery "Part9"
    DoCmd.Close acQuery, "Part9"
    DoCmd.OpenQuery "Defect4"
    DoCmd.Close acQuery, "Defect4"
    DoCmd.OpenQuery "Part10"
    DoCmd.Close acQuery, "Part10"
    DoCmd.OpenQuery "Defect5"
    DoCmd.Close acQuery, "Defect5"
    Application.SysCmd acSysCmdUpdateMeter, 400
    DoCmd.OpenQuery "Defect6"
    DoCmd.Close acQuery, "Defect6"
    DoCmd.OpenQuery "Part11"
    DoCmd.Close acQuery, "Part11"
    DoCmd.OpenQuery "Part12"
    DoCmd.Close acQuery, "Part12"
    DoCmd.OpenQuery "Defect7"
DoCmd.Close acQuery, "Defect7"
DoCmd.OpenQuery "Defect8"
DoCmd.Close acQuery, "Defect8"
Application.SysCmd acSysCmdUpdateMeter, 500
DoCmd.OpenQuery "Defect9"
DoCmd.Close acQuery, "Defect9"
DoCmd.OpenQuery "Defect10"
DoCmd.Close acQuery, "Defect10"
DoCmd.OpenQuery "Part13"
DoCmd.Close acQuery, "Part13"
DoCmd.OpenQuery "Part14"
DoCmd.Close acQuery, "Part14"
DoCmd.OpenQuery "Part15"
DoCmd.Close acQuery, "Part15"
DoCmd.OpenQuery "Defect11"
DoCmd.Close acQuery, "Defect11"
Application.SysCmd acSysCmdUpdateMeter, 600
DoCmd.OpenQuery "Defect12"
DoCmd.Close acQuery, "Defect12"
DoCmd.OpenQuery "Special1"
DoCmd.Close acQuery, "Special2"
DoCmd.OpenQuery "Defect13"
DoCmd.Close acQuery, "Defect13"
DoCmd.OpenQuery "Special2"
DoCmd.Close acQuery, "Special2"
Application.SysCmd acSysCmdUpdateMeter, 700
DoCmd.OpenQuery "Part16"
DoCmd.Close acQuery, "Part16"
DoCmd.OpenQuery "Special3"
DoCmd.Close acQuery, "Special3"
DoCmd.OpenQuery "Part17"
DoCmd.Close acQuery, "Part17"
DoCmd.OpenQuery "Part18"
DoCmd.Close acQuery, "Part18"
Application.SysCmd acSysCmdUpdateMeter, 800
DoCmd.OpenQuery "Part19"
DoCmd.Close acQuery, "Part19"
DoCmd.OpenQuery "Part20"
DoCmd.Close acQuery, "Part20"
DoCmd.OpenQuery "Part21"
DoCmd.Close acQuery, "Part21"
DoCmd.OpenQuery "Part22"
DoCmd.Close acQuery, "Part22"
DoCmd.OpenQuery "Defect14"
DoCmd.Close acQuery, "Defect14"
Application.SysCmd acSysCmdUpdateMeter, 900
DoCmd.OpenQuery "Part23"
DoCmd.Close acQuery, "Part23"
DoCmd.OpenQuery "Defect15"
DoCmd.Close acQuery, "Defect15"
DoCmd.OpenQuery "Part24"
DoCmd.Close acQuery, "Part24"
DoCmd.OpenQuery "Defect16"
Application.SysCmd acSysCmdUpdateMeter, 1000
DoCmd.Close acQuery, "Defect16"
```

```
    DoCmd.OpenQuery "Special4"
    DoCmd.Close acQuery, "Special4"
    Application.SysCmd acSysCmdRemoveMeter 'removes progress bar

End Sub
```

```
Sub makeupdatedtable()
'****************************************************************************************************
'****************************************************************************************************
'*******                                                                                    *********
'*******    This function makes the updated table which combines all of the information     *********
'*******    from the identifier queries into a single table                                 *********
'*******                                                                                    *********
'****************************************************************************************************
'****************************************************************************************************
'*******  Created by Jeremy Lieberman   *******  December 9, 2011   *******   V1.0   ****************
'****************************************************************************************************

    Application.SysCmd acSysCmdInitMeter, "Part 2: Creating Updated Data Table", 7000
                                             'display progress bar


    DoCmd.OpenQuery "Clear Updated Data"
    DoCmd.OpenQuery "Create_Updated_Data_Table" 'most of the columns are created by this query

    'the remaining code calls the add queries which update the table so that each record will include
    'notes of all of the part and defect codes that apply for that record, aiding analysis of the data
    DoCmd.OpenQuery "Process_Step_Add_Flag_1"
    DoCmd.OpenQuery "Process_Step_Add_Flag_2"
    DoCmd.OpenQuery "Process_Step_Add_Flag_3"
    DoCmd.OpenQuery "Process_Step_Add_Flag_4"
    Application.SysCmd acSysCmdUpdateMeter, 1000
    DoCmd.OpenQuery "Process_Step_Add_Flag_5"
    DoCmd.OpenQuery "Process_Step_Add_Flag_6"
    DoCmd.OpenQuery "Process_Step_Add_Flag_7"
    DoCmd.OpenQuery "Process_Step_Add_Flag_8"
    DoCmd.OpenQuery "Process_Step_Add_Flag_9"
    Application.SysCmd acSysCmdUpdateMeter, 2000
    DoCmd.OpenQuery "Process_Step_Add_Flag_10"
    DoCmd.OpenQuery "Process_Step_Add_Flag_11"
    DoCmd.OpenQuery "Process_Step_Add_Flag_12"
    DoCmd.OpenQuery "Process_Step_Add_Flag_13"
    DoCmd.OpenQuery "Process_Step_Add_Flag_14"
    DoCmd.OpenQuery "Process_Step_Add_Flag_15"
    Application.SysCmd acSysCmdUpdateMeter, 3000
    DoCmd.OpenQuery "Process_Step_Add_Flag_16"
    DoCmd.OpenQuery "Process_Step_Add_Flag_17"
    DoCmd.OpenQuery "Process_Step_Add_Flag_18"
    DoCmd.OpenQuery "Process_Step_Add_Flag_19"
    DoCmd.OpenQuery "Process_Step_Add_Flag_20"
    Application.SysCmd acSysCmdUpdateMeter, 4000
    DoCmd.OpenQuery "Process_Step_Add_Flag_21"
    DoCmd.OpenQuery "Process_Step_Add_Flag_22"
    DoCmd.OpenQuery "Process_Step_Add_Flag_23"
    DoCmd.OpenQuery "Process_Step_Add_Flag_24"
    DoCmd.OpenQuery "Process_Step_Add_Flag_25"
    DoCmd.OpenQuery "Process_Step_Add_Flag_26"
    DoCmd.OpenQuery "Process_Step_Add_Flag_27"
    Application.SysCmd acSysCmdUpdateMeter, 5000
    DoCmd.OpenQuery "Process_Step_Add_Flag_28"
    DoCmd.OpenQuery "Process_Step_Add_Flag_29"
    DoCmd.OpenQuery "Process_Step_Add_Flag_30"
    DoCmd.OpenQuery "Process_Step_Add_Flag_31"
    DoCmd.OpenQuery "Process_Step_Add_Flag_32"
    DoCmd.OpenQuery "Process_Step_Add_Flag_33"
    Application.SysCmd acSysCmdUpdateMeter, 6000
    DoCmd.OpenQuery "Process_Step_Add_Flag_34"
    DoCmd.OpenQuery "Process_Step_Add_Flag_35"
    DoCmd.OpenQuery "Process_Step_Add_Flag_36"
    DoCmd.OpenQuery "Process_Step_Add_Flag_37"
    DoCmd.OpenQuery "Process_Step_Add_Flag_38"
    DoCmd.OpenQuery "Process_Step_Add_Flag_39"
    DoCmd.OpenQuery "Process_Step_Add_Flag_40"
    Application.SysCmd acSysCmdUpdateMeter, 7000
    DoCmd.OpenQuery "Process_Step_Add_Flag_41"
    DoCmd.OpenQuery "Process_Step_Add_Flag_42"
    DoCmd.OpenQuery "Process_Step_Add_Flag_43"
    DoCmd.OpenQuery "Process_Step_Add_Flag_44"
    DoCmd.OpenQuery "Process_Step_Add_Flag_45"
    Call Add_Week
    Application.SysCmd acSysCmdRemoveMeter 'removes progress bar

End Sub
```

```
Sub defectvspartpivot()
'**************************************************************************************************
'**************************************************************************************************
'*******                                                                                  *********
'*******    This function makes the defect vs part pivot table                            *********
'*******                                                                                  *********
'**************************************************************************************************
'**************************************************************************************************
'*******  Created by Jeremy Lieberman   *******  December 9, 2011   *******   V1.0   ***************
'**************************************************************************************************

    Application.SysCmd acSysCmdInitMeter, "Part 3: Creating Defect vs part pivot", 800
                                               'display progress bar
    DoCmd.OpenQuery "Clear_Defect_vs_Part"     'resets the table before updating the data
    DoCmd.OpenQuery "Generate_Row_1"
    DoCmd.OpenQuery "Generate_Row_2"
    Application.SysCmd acSysCmdUpdateMeter, 100
    DoCmd.OpenQuery "Generate_Row_3"
    DoCmd.OpenQuery "Generate_Row_4"
    DoCmd.OpenQuery "Generate_Row_5"
    DoCmd.OpenQuery "Generate_Row_6"
    Application.SysCmd acSysCmdUpdateMeter, 200
    DoCmd.OpenQuery "Generate_Row_7"
    DoCmd.OpenQuery "Generate_Row_8"
    DoCmd.OpenQuery "Generate_Row_9"
    Application.SysCmd acSysCmdUpdateMeter, 300
    DoCmd.OpenQuery "Generate_Row_10"
    DoCmd.OpenQuery "Generate_Row_11"
    DoCmd.OpenQuery "Generate_Row_12"
    Application.SysCmd acSysCmdUpdateMeter, 400
    DoCmd.OpenQuery "Generate_Row_13"
    DoCmd.OpenQuery "Generate_Row_14"
    DoCmd.OpenQuery "Generate_Row_15"
    Application.SysCmd acSysCmdUpdateMeter, 500
    DoCmd.OpenQuery "Generate_Row_16"
    DoCmd.OpenQuery "Generate_Row_17"
    DoCmd.OpenQuery "Generate_Row_18"
    Application.SysCmd acSysCmdUpdateMeter, 600
    DoCmd.OpenQuery "Generate_Row_19"
    DoCmd.OpenQuery "Generate_Row_20"
    DoCmd.OpenQuery "Generate_Row_21"
    Application.SysCmd acSysCmdUpdateMeter, 700
    DoCmd.OpenQuery "Generate_Row_22"
    DoCmd.OpenQuery "Generate_Row_23"
    DoCmd.OpenQuery "Generate_Row_24"
    Application.SysCmd acSysCmdUpdateMeter, 800
    DoCmd.OpenQuery "Generate_Row_25"
    DoCmd.OpenQuery "Generate_Row_26"
    DoCmd.OpenQuery "Generate_Total_Row"

    Application.SysCmd acSysCmdRemoveMeter 'removes progress bar
End Sub
```

C.2.2 Other Functions

One function is included to give an indication of some of the capabilities of the tool. Ultimately, these functions are meant to provide customization to the tool when different datasets are processed. As a result, these functions will need to be developed by each new user wishing to add features beyond the creation of the basic heat map.

```
Public Sub Add Defect and Part Count()
'*****************************************************************************************
'*****************************************************************************************
'*******                                                                        *********
'*******    This function totals the number of defect and part descriptions that apply  *********
'*******    to each record.  It then updates the columns in the record to provide        *********
'*******    the total for defects, parts and both types of descriptions                 *********
'*******                                                                        *********
'*****************************************************************************************
'*****************************************************************************************
'******* Created by Jeremy Lieberman  ******* December 9, 2011  *******  V1.0  ****************
'*****************************************************************************************

'******************************      Variable Declaration      ***************************************
```

```
Dim dbtemp As DAO.Database  'temporary store the database for this function
Dim rstdata As DAO.Recordset 'temporary store the updated_data table for this function
Dim defect As Integer, part As Integer, total As Integer

Set dbtemp = CurrentDb
Set rstdata = dbtemp.OpenRecordset("Updated Data", dbOpenTable)

'******************************     Primary Code            ****************************************

rstdata.MoveFirst

Do While rstdata.EOF = False

    defect = 0
    part = 0     'resets the values to 0
    total = 0

    'the next section of code counts if any of the defect fields are checked
    If rstdata!Defect1 = "X" Then defect = defect + 1
    If rstdata!Defect2 = "X" Then defect = defect + 1
    If rstdata!Defect3 = "X" Then defect = defect + 1
    If rstdata!Defect4 = "X" Then defect = defect + 1
    If rstdata!Defect5 = "X" Then defect = defect + 1
    If rstdata!Defect6 = "X" Then defect = defect + 1
    If rstdata!Defect7 = "X" Then defect = defect + 1
    If rstdata!Defect8 = "X" Then defect = defect + 1
    If rstdata!Defect9 = "X" Then defect = defect + 1
    If rstdata!Defect10 = "X" Then defect = defect + 1
    If rstdata!Defect11 = "X" Then defect = defect + 1
    If rstdata!Defect12 = "X" Then defect = defect + 1
    If rstdata!Defect13 = "X" Then defect = defect + 1
    If rstdata!Defect14 = "X" Then defect = defect + 1
    If rstdata!Void = "X" Then defect = defect + 1
    If rstdata!Wrongsized = "X" Then defect = defect + 1

    'the next section of code counts if any of the parts are checked
    If rstdata!Part1 = "X" Then part = part + 1
    If rstdata!Part2 = "X" Then part = part + 1
    If rstdata!Part3 = "X" Then part = part + 1
    If rstdata!Part4 = "X" Then part = part + 1
    If rstdata!Part5 = "X" Then part = part + 1
    If rstdata!Part6 = "X" Then part = part + 1
    If rstdata!Part7 = "X" Then part = part + 1
    If rstdata!Part8 = "X" Then part = part + 1
    If rstdata!Part9 = "X" Then part = part + 1
    If rstdata!Part10 = "X" Then part = part + 1
    If rstdata!Part11 = "X" Then part = part + 1
    If rstdata!Part12 = "X" Then part = part + 1
    If rstdata!Part13 = "X" Then part = part + 1
    If rstdata!Part14 = "X" Then part = part + 1
    If rstdata!Part15 = "X" Then part = part + 1
    If rstdata!Part16 = "X" Then part = part + 1
    If rstdata!Part17 = "X" Then part = part + 1
    If rstdata!Part18 = "X" Then part = part + 1
    If rstdata!Part19 = "X" Then part = part + 1
    If rstdata!Part20 = "X" Then part = part + 1
    If rstdata!Part21 = "X" Then part = part + 1
    If rstdata!Part22 = "X" Then part = part + 1
    If rstdata!Part23 = "X" Then part = part + 1
    If rstdata!Part24 = "X" Then part = part + 1
    If rstdata!Part25 = "X" Then part = part + 1

    total = defect + part 'computes the total

    'the next loop updates the fields in the table to reflect the totals
    With rstdata
    .Edit
    ![Number of Defects] = defect
    ![Number of Parts] = part
    ![Total Defects and Parts] = total
    .Update
    End With

    rstdata.MoveNext

Loop

rstdata.Close
dbtemp.Close

End Sub
```

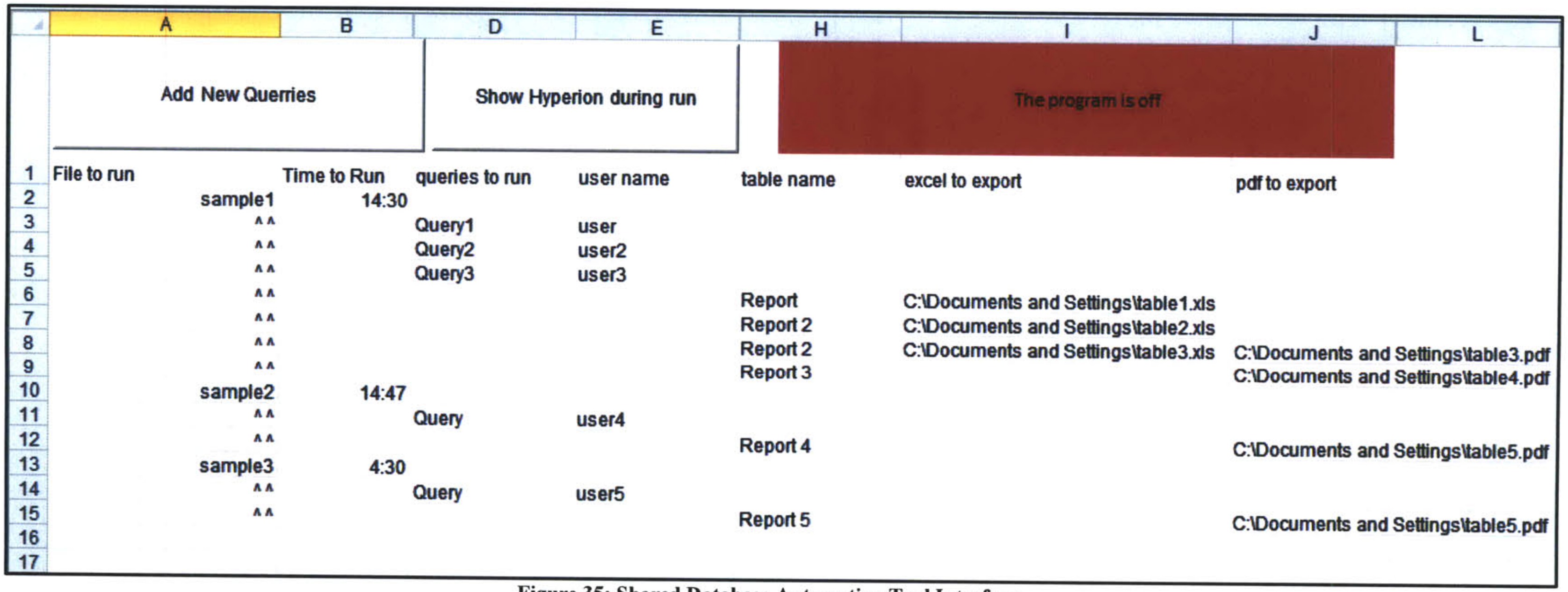
Appendix D

D.1 Shared Database Automation Tool Summary

The Database Automation Tool was developed with a Microsoft Excel interface utilizing VBA to provide all of the program's functionality. Unlike the Word Parser Tool, the Database Automation was created for widespread use and would need to have an interface that would be very accessible to employees with a greater range in computer experience. To this end, a Microsoft Excel interface was chosen due to its prevalence and ease of operation. A full size image of the interface is included in Section D.2 as Figure 35.

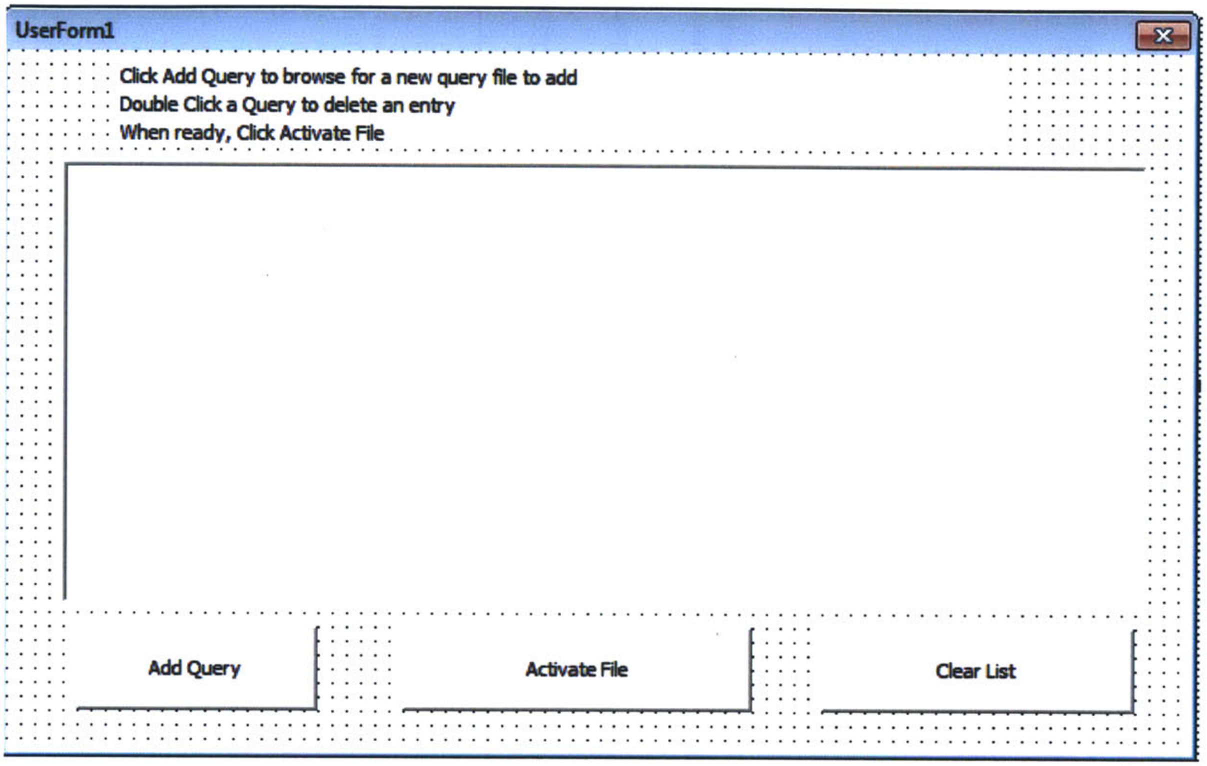
When a user decides to schedule a new query, the program provides a series of input boxes walking him/her through all of the information needed to process the file. As each of these boxes is completed, the program constructs a program in javascript that will instruct the database to export the desired data. A copy of these input boxes is included in Section D.2. Once the data has been exported, an additional part of the program can automate distribution of these files. This part of the program has not been included in this thesis due to proprietary concerns.

D.2 Shared Database Automation Tool Interface

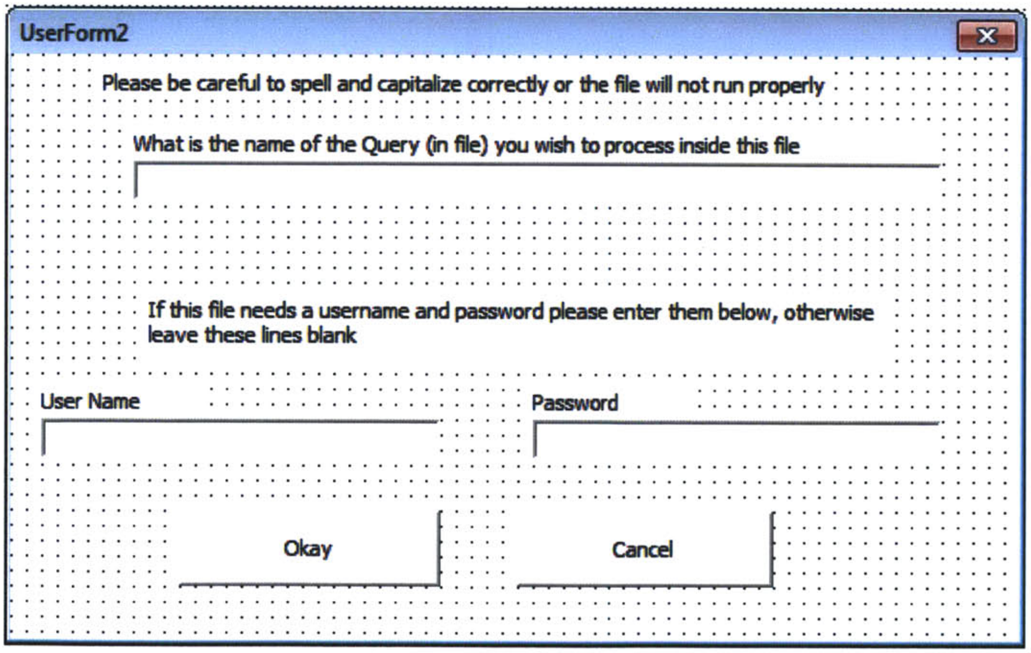


	A	B	D	E	H	I	J
1	File to run	Time to Run	queries to run	user name	table name	excel to export	pdf to export
2	sample1	14:30					
3	^ ^		Query1	user			
4	^ ^		Query2	user2			
5	^ ^		Query3	user3			
6	^ ^				Report	C:\Documents and Settings\table1.xls	
7	^ ^				Report 2	C:\Documents and Settings\table2.xls	
8	^ ^				Report 2	C:\Documents and Settings\table3.xls	C:\Documents and Settings\table3.pdf
9	^ ^				Report 3		C:\Documents and Settings\table4.pdf
10	sample2	14:47					
11	^ ^		Query	user4			
12	^ ^				Report 4		C:\Documents and Settings\table5.pdf
13	sample3	4:30					
14	^ ^		Query	user5			
15	^ ^				Report 5		C:\Documents and Settings\table5.pdf
16							
17							

Figure 35: Shared Database Automation Tool Interface



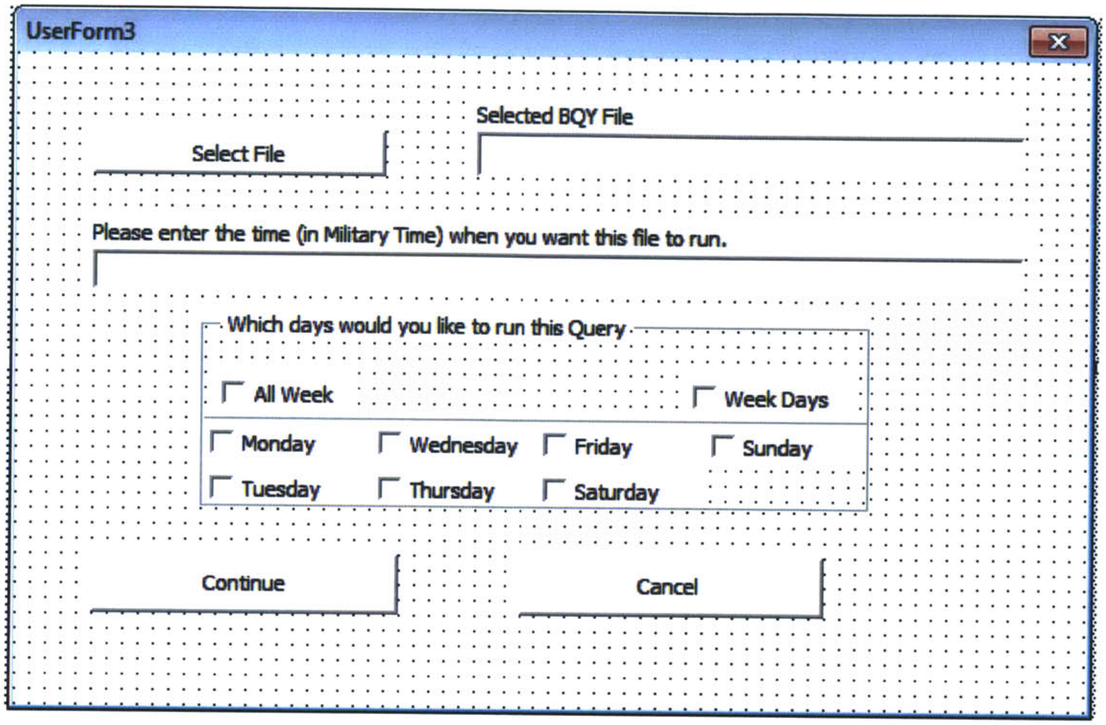
Click Add Query to browse for a new query file to add
Double Click a Query to delete an entry
When ready, Click Activate File



Please be careful to spell and capitalize correctly or the file will not run properly

What is the name of the Query (in file) you wish to process inside this file

If this file needs a username and password please enter them below, otherwise leave these lines blank



UserForm3

Select File

Selected BQY File

Please enter the time (in Military Time) when you want this file to run.

Which days would you like to run this Query

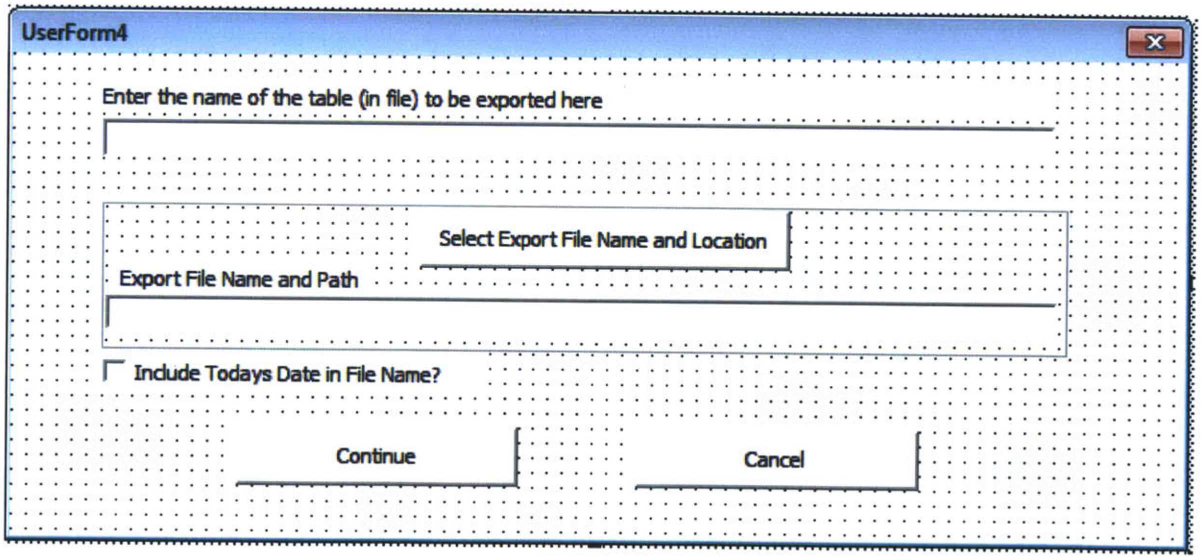
□ All Week □ Week Days

□ Monday □ Wednesday □ Friday □ Sunday

□ Tuesday □ Thursday □ Saturday

Continue

Cancel



UserForm4

Enter the name of the table (in file) to be exported here

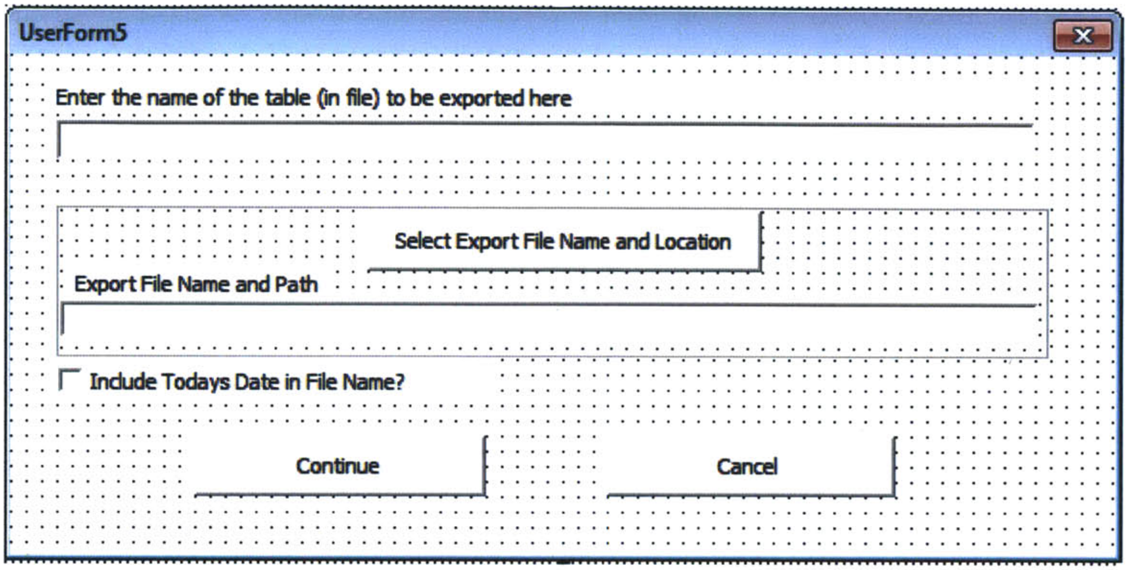
Select Export File Name and Location

Export File Name and Path

□ Include Todays Date in File Name?

Continue

Cancel


UserForm5
Enter the name of the table (in file) to be exported here
Select Export File Name and Location
Export File Name and Path
□ Include Todays Date in File Name?
Continue
Cancel

D.3 Shared Database Automation Tool Code

The VBA code which controls the interface is divided into workbook and worksheet pages, 5 userforms, and 4 modules. The workbook and worksheet pages contain code that is triggered when certain actions occur in either the workbook or worksheet (e.g. the file is opened or certain cells are double clicked). Each of the userforms, corresponds to the input boxes included in the previous section. The remaining code was split into different modules based upon the function that it would perform.

D.3.1 Workbook and Worksheet Code

The code in the workbook and worksheet's includes the following.

- Workbook_Open
- Worksheet_BeforeDoubleClick

```
Private Sub Workbook Open()
'****************************************************************************************************
'****************************************************************************************************
'*******                                                                                    *********
'*******    This code runs when the program is opened.  It is intended to lock the file     *********
'*******    and preset the global variables so that the file is ready for use               *********
'*******                                                                                    *********
'****************************************************************************************************
'****************************************************************************************************
'*******  Created by Jeremy Lieberman   *******  December 13, 2011   *******  V1.2   **************
'****************************************************************************************************

Dim stcur As Worksheet

Set shtcur = ThisWorkbook.Sheets("Auto Home Page")

shtcur.Unprotect ("Lieberman")  'turns off the protection on the tab for presetting variables

Call nosplash                    'runs the nosplash sub to set the parameter, other code page
Call deactivatecode             'runs the deactivate code to preset the file to deactivated
runtimer = 0                    'resets the run timer

shtcur.Protect ("Lieberman")    'turns on the protection for the sheet

End Sub
```

```vb
'Functions on the Auto_Home_Page tab are intended to respond to key actions on that page
'In this case, when a user double clicks on a cell it will determine if any follow up action is desired

Private Sub Worksheet_BeforeDoubleClick(ByVal Target As Range, Cancel As Boolean)
'*******************************************************************************************
'*******************************************************************************************
'*******   This function is activated when the user double clicks on a cell on the main    ********
'*******   screen and is intended to allow the user to update certain cells without having ********
'*******   to remove the security features, thus protecting against bad edits              ********
'*******************************************************************************************
'*******************************************************************************************
'******* Created by Jeremy Lieberman   *******  December 13, 2011   *******   V1.2   *************
'*******************************************************************************************

'****************************       Variable Declaration       ***********************************

Dim proceed2 As String      'variable to determine if the user canceled out of the file select
Dim proceed As String       'variable holds answer to the user prompt for permission
Dim stcur As Worksheet      'auto home page tab
Dim code_col As Integer     'stores the column which holds the database' s controlling code

'****************************       Initial Variable Set       ***********************************

Set shtcur = ThisWorkbook.Sheets("Auto Home Page")          '
shtcur.Unprotect ("Lieberman")                              'unprotect file for editing


proceed = vbNo  'preset the first user proceed choice to don't proceed
proceed2 = ""   'preset the second user proceed choice to don't proceed

'****************************       Primary Code               ***********************************

'the next piece of code enables the user to update the file name and/or location
If ActiveCell.Column = 1 And ActiveCell.Row > 1 Then         'location of the file name and location
    If ActiveCell.Value <> "" And ActiveCell.Value <> "ˆ ˆ" Then
        proceed = MsgBox("Do you want to change this file name and/or location?", vbYesNo, "Proceed?")
    End If

    If proceed = vbYes Then
        proceed2 = Application.GetOpenFilename("*.XXX, *.XXX")
            If proceed2 <> "" And proceed2 <> "False" Then  'conditions if the user cancels the input
                ActiveCell = proceed2                       'new location of the file
                code_col = Cells(ActiveCell.Row, 11).Value  'location of the database script in excel
                proceed2 = Replace(proceed2, "\", "\\")
                Cells(2, code_col) = "Application.Documents.Open(" & Chr(34) & proceed2 & Chr(34) & ")"
                                                            'updates the database script

            End If
    End If
End If


' The next piece of code enables the user to update the time of run
If ActiveCell.Column = 2 And ActiveCell.Row > 1 Then      'location of the time when each file will run
    If ActiveCell.Value <> "" Then
        proceed = MsgBox("Do you want to change when this file will run?", vbYesNo, "Proceed?")
    End If
    If proceed = vbYes Then
        proceed2 = Application.InputBox("Please enter the time (in Military Time) when you want this _
                                        file to run.", "Pick Time")

        If proceed2 <> "" And proceed2 <> "False" Then
            ActiveCell.Value = proceed2
            MsgBox ("To complete this change, relaunch (close and reopen) the Auto Excel")
        End If
    End If

End If

'the next piece of code enables the user to update the day that the code will run
If ActiveCell.Column = 7 And ActiveCell.Row > 1 Then     'contains the date that the file is being run
    If ActiveCell.Value <> "" Then
        proceed = MsgBox("Do you want to change the days this file will run?", vbYesNo, "Proceed?")
    End If

    If proceed = vbYes Then
        UserForm6.Show
    End If
End If

ThisWorkbook.Save

Cancel = True                   'ensures this is the only action that occurs
shtcur.Protect ("Lieberman")    'reprotect the file
End Sub
```

D.3.2 Userform Code

The userforms control the interface for adding new queries to the automation file. A copy of the forms is included in section D.2. The following sections contain the code that controls the six forms.

D.3.2.1 Userform 1

This userform creates the primary interface for entering and removing programs from the automation file.

```
Private Sub Userform_activate()

'**************************************************************************************************
'**************************************************************************************************
'*******                                                                                 *********
'*******    This function is launched when the user clicks the select file button        *********
'*******    The file loads existing data from the excel file and drops it into the       *********
'*******    userform information box for review by the user.  It also allows the user    *********
'*******    to activate the auto maker and delete queries that have been scheduled       *********
'*******                                                                                 *********
```

```
'**************************************************************************************************
'**************************************************************************************************
'*******  Created by Jeremy Lieberman  *******  December 13, 2011  *******  V1.0  **************
'**************************************************************************************************

'******************************     Variable Declaration      *************************************

Dim filename As String      'stores the name of scheduled files
Dim Timestart As String     'stores the time when the file will operate
Dim Query As String         'stores the query name that will run
Dim user As String          'stores the username for the query being loaded
Dim table As String         'stores the table that will be exported
Dim excel_export As String  'stores excels that will be exported
Dim pdf_export As String     'stores pdfs that will be exported
Dim lastrun As Integer      'determines the last row that contains information
Dim shtcur As Worksheet     'auto home page tab

'******************************     Primary Code       *********************************************

Set shtcur = ThisWorkbook.Sheets("Auto Home Page")

shtcur.Unprotect ("Lieberman")       'turns off security on file to enable processing

Sheets("Auto Home Page").Activate
lastrun = Range("A65536").End(xlUp).Row


'the following loop populates the listbox within the userform displaying all of the active queries
'it is formatted to use line separators and tabbed spacing to provide a clean and easy to read
'appearance for the user
For x = 2 To lastrun
    If Cells(x, 1) <> "" Then
        If Cells(x, 1) = "ˆ ˆ" Then
            filename = ""
        Else
            filename = cutpath(Cells(x, 1).text)    'pulls the name of existing files into the list
        End If
        Timestart = Cells(x, 2).text                'pulls the time to run into the list
        Query = Cells(x, 4).text                    'pulls the name of the queries
        user = Cells(x, 5).text                     'pulls the username for each query
        table = Cells(x, 8).text                    'pulls the name of any tables to be exported
        excel_export = cutpath(Cells(x, 9).text)    'pulls the name of any export excel files
        pdf_export = cutpath(Cells(x, 10).text)     'pulsl teh name of any export pdfs
        ListBox1.AddItem (filename & "         " & Timestart & "    " & Query & "    " & user & _
                    "    " & table & "    " & excel_export & "    " & pdf_export)
'load each line of information into the list box
    End If
Next
End Sub
```

```
Sub AddQuery_Click()
'***********************************************************************************************
'***********************************************************************************************
'*******                                                                              *********
'*******    This function is linked to the add query button within the first user form.    *********
'*******    When combined with other userforms, it guides the user through entering         *********
'*******    all of the information needed to schedule a report.  It will also create the    *********
'*******    javascript needed to control the database accessing tools.                       *********
'*******                                                                              *********
'***********************************************************************************************
'***********************************************************************************************
'*******  Created by Jeremy Lieberman   *******  December 13, 2011   *******   V1.0   **************
'***********************************************************************************************

'This userform will create all of the code that will be generated so that it does not need
'to pass the code_colnum variable into each of the userforms.  As a result, after each userform puts
'its results into the excel file, that information will than be picked up again by this form.  This
'saves the use of additional global variable for each other userfrom to accept this information.

'*****************************     Variable Declaration      *************************************

Dim code_colnum As Integer              'This variable stores which column will hold this file's code
Dim lastrun As Integer                  'last file in the automation code
Dim filerow As Integer                  'row that stores the next file
Dim codebuilder As Integer              'for creating the java instructions
Dim i As Integer                        'parsing variable
Dim filename As String
Dim FileNameforTool As String       'name of database accessing file
Dim user As String                      'username
Dim pass As String                      'password
Dim Query As String                     'query within database file
Dim tableexport As String               'table to be exported
Dim Excelexport As String               'exported excel file
Dim PDFexport As String                 'exported pdf file
Dim NumQuery As Integer                 'number of queries to process in the file
Dim numexcel As Integer                 'number of excel files to be exported
Dim numpdf As Integer                   'number of pdfs to be exported

ThisWorkbook.Sheets("Auto Home Page").Select

'*****************************     Primary Code        ********************************************

'The first part of the code determines where critical information will be stored in the excel
spreadsheet
filerow = Range("A65536").End(xlUp).Row + 1 'records the row where the next data file will be stored
code_colnum = Range("XFD2").End(xlToLeft).Column + 1 'location where the java code will be stored
Columns(code_colnum).ClearContents ' wipes the column of any anomalous data that is stored there


'The next part of the code opens the first userform for inputting basic query information.
' This userform will drop its results into the spreadsheet
UserForm3.filrow = filerow      'this passes in a key piece of information to the userform
UserForm3.Show                  'activate the form

filename = Cells(filerow, 1)
```

```
If filename = "" Then GoTo 1 'abort add query if no file is selected
FileNameforTool = Replace(filename, "\", "\\")
codebuilder = 2
Cells(codebuilder, code_colnum) = "Application.Documents.Open(" & Chr(34) & FileNameforTool & Chr(34) &
                                                            ")"
codebuilder = codebuilder + 1

' ***************************************************************************************

'the next part of the code controls the input for building the queries within the file that will be run
'first it determines how many queries the user wants to run in the file and then has the user input
'the instructions for each query through userform 2

NumQuery = Application.InputBox("How many Queries do you want to process in this file?", , , , , , , 1)
If NumQuery = False Or NumQuery = 0 Then GoTo 1 'aborts the file on cancel press

Cells(filerow, 11) = code_colnum    'stores the column where the javascript will be stored
filerow = filerow + 1
For i = 1 To NumQuery
        UserForm2.i = i
        UserForm2.filrow = filerow  'inputs and stores information regarding the query
        UserForm2.Show
        If Cells(filerow, 4) = "" Then
        MsgBox ("Query " & i & " was cancelled")
        Else
            If Cells(filerow, 5) <> "" Then
                Query = Cells(filerow, 4)   'pulls the query name from the excel for use
                user = Cells(filerow, 5)    'pulls the username from the excel for use
                pass = Cells(filerow, 6)    'pulls the password from the excel for use

                'the next section of code writes the javascript and stores it in the excel
                Cells(codebuilder, code_colnum) = "ActiveDocument.Sections[" & Chr(34) & Query & _
                          Chr(34) & "].DataModel.Connection.Username = " & Chr(34) & user & Chr(34)

                codebuilder = codebuilder + 1

                Cells(codebuilder, code_colnum) = "ActiveDocument.Sections[" & Chr(34) & Query & _
                        Chr(34) & "].DataModel.Connection.SetPassword(" & Chr(34) & pass & Chr(34) &
                                                                     ")"

                codebuilder = codebuilder + 1

                Cells(codebuilder, code_colnum) = "ActiveDocument.Sections[" & Chr(34) & Query & _
                                                  Chr(34) & "].DataModel.Connection.Connect()"

                codebuilder = codebuilder + 1
            End If

        Query = Cells(filerow, 4)
        Cells(codebuilder, code_colnum) = "ActiveDocument.Sections[" & Chr(34) & Query & Chr(34) & _
                                          "].Process()"
        codebuilder = codebuilder + 1
        If (Cells(filerow, 1).Value = "") Then Cells(filerow, 1).Value = "^ ^"
        filerow = filerow + 1

        End If
Next

' ******************************************************************************************
' the following section of code creates the excel exports for each file

numexcel = Application.InputBox("How many excel tables do you want to process?", , , , , , , 1)
If numexcel = False Or numexcel = 0 Then GoTo 2 'skip section

For i = 1 To numexcel
    UserForm4.filrow = filerow  'excel export userform, stores information in the excel
    UserForm4.Show
        If Cells(filerow, 8) = "" Then
            MsgBox ("export " & i & " was cancelled")
        Else

            tableexport = Cells(filerow, 8) 'pulls the table name after the file have been added
            Excelexport = Cells(filerow, 9) 'pulls the excel name for creating the code

            Excelexport = Replace(Excelexport, "\", "\\")

            Cells(codebuilder, code_colnum) = "ActiveDocument.Sections[" & Chr(34) & tableexport &
                Chr(34) & "].Export(" & Chr(34) & Excelexport & Chr(34) & ",bqExportFormatExcel2,1,0)"
            codebuilder = codebuilder + 1

            If (Cells(filerow, 1).Value = "") Then Cells(filerow, 1).Value = "^ ^"
            filerow = filerow + 1

        End If
Next

2

' ******************************************************************************************
'the next section of code creates the pdf exports for each file

numpdf = Application.InputBox("How many pdf files do you want to process in this file?", , , , , , , 1)
If numpdf = False Then GoTo 3 'skip section

For i = 1 To numpdf
    UserForm5.filrow = filerow
    UserForm5.Show
        If Cells(filerow, 8) = "" Then
            MsgBox ("export " & i & " was cancelled")
        Else

            tableexport = Cells(filerow, 8)
            PDFexport = Cells(filerow, 10)

            PDFexport = Replace(PDFexport, "\", "\\")

            Cells(codebuilder, code_colnum) = "ActiveDocument.Sections[" & Chr(34) & tableexport &
                Chr(34) & "].Export(" & Chr(34) & PDFexport & Chr(34) & ",bqExportFormatPDF,1,0)"
            codebuilder = codebuilder + 1

            If (Cells(filerow, 1).Value = "") Then Cells(filerow, 1).Value = "^ ^"
            filerow = filerow + 1

        End If
Next

3

'the final section of code adds the final javascript instructions to the file closing out the database
'accessing function

    Cells(codebuilder, code_colnum) = "var myPath=""C:\\path\\script.txt"" "
    codebuilder = codebuilder + 1
    Cells(codebuilder, code_colnum) = "var oleApp = new JOOLEObject(""Scripting.FileSystemObject"")"
    codebuilder = codebuilder + 1
    Cells(codebuilder, code_colnum) = "var traceDoc=oleApp.CreateTextFile(myPath)"
    codebuilder = codebuilder + 1
    Cells(codebuilder, code_colnum) = "traceDoc.Close()"
    codebuilder = codebuilder + 1
    Cells(codebuilder, code_colnum) = "Application.Quit()"
    codebuilder = codebuilder + 1
    Unload Me
    UserForm1.Show

1
End Sub
```

```
Private Sub ListBox1_DblClick(ByVal Cancel As MSForms.ReturnBoolean)
'**************************************************************************************************
'**************************************************************************************************
'*******                                                                                  *********
'*******    This function deletes a specific file from the automation file.  It is        *********
'*******    activated when the user double clicks on a line in the list box and           *********
'*******    includes a confirmation to proceed                                            *********
'*******                                                                                  *********
'**************************************************************************************************
'**************************************************************************************************
'*******  Created by Jeremy Lieberman   *******  December 13, 2011   *******   V1.0   ***************
'**************************************************************************************************

Dim offset As Integer
Dim code As Integer


Response = MsgBox(prompt:="Delete?: 'Yes' or 'No'.", Buttons:=vbYesNo)  'check for confirmation

If Response = vbYes Then
    Sheets("Auto Home Page").Activate
    offset = 0
    'the next loop locates the first line in the file that is going to be deleted
    Do While (Cells(ListBox1.ListIndex + offset + 2, 1) = "^ ^")
        offset = offset - 1
    Loop

    code = Cells(ListBox1.ListIndex + offset + 2, 11).Value 'delete the first row of the file
    Cells(ListBox1.ListIndex + offset + 2, 1).Delete shift:=xlUp
    Cells(ListBox1.ListIndex + offset + 2, 2).Delete shift:=xlUp
    Cells(ListBox1.ListIndex + offset + 2, 3).Delete shift:=xlUp
    Cells(ListBox1.ListIndex + offset + 2, 4).Delete shift:=xlUp
    Cells(ListBox1.ListIndex + offset + 2, 5).Delete shift:=xlUp
    Cells(ListBox1.ListIndex + offset + 2, 6).Delete shift:=xlUp
    Cells(ListBox1.ListIndex + offset + 2, 7).Delete shift:=xlUp
    Cells(ListBox1.ListIndex + offset + 2, 8).Delete shift:=xlUp
    Cells(ListBox1.ListIndex + offset + 2, 9).Delete shift:=xlUp
    Cells(ListBox1.ListIndex + offset + 2, 10).Delete shift:=xlUp
    Cells(ListBox1.ListIndex + offset + 2, 11).Delete shift:=xlUp
    Cells(ListBox1.ListIndex + offset + 2, 12).Delete shift:=xlUp

    'the following loop deletes all of the rows until the next file begins
    Do While Cells(ListBox1.ListIndex + offset + 2, 1).text = "^ ^"
        Cells(ListBox1.ListIndex + offset + 2, 1).Delete shift:=xlUp
        Cells(ListBox1.ListIndex + offset + 2, 2).Delete shift:=xlUp
        Cells(ListBox1.ListIndex + offset + 2, 3).Delete shift:=xlUp
        Cells(ListBox1.ListIndex + offset + 2, 4).Delete shift:=xlUp
        Cells(ListBox1.ListIndex + offset + 2, 5).Delete shift:=xlUp
        Cells(ListBox1.ListIndex + offset + 2, 6).Delete shift:=xlUp
        Cells(ListBox1.ListIndex + offset + 2, 7).Delete shift:=xlUp
        Cells(ListBox1.ListIndex + offset + 2, 8).Delete shift:=xlUp
        Cells(ListBox1.ListIndex + offset + 2, 9).Delete shift:=xlUp
        Cells(ListBox1.ListIndex + offset + 2, 10).Delete shift:=xlUp
        Cells(ListBox1.ListIndex + offset + 2, 11).Delete shift:=xlUp
        Cells(ListBox1.ListIndex + offset + 2, 12).Delete shift:=xlUp
    Loop

Columns(code).ClearContents 'removes the java code for the file

Unload Me   'reset the menu
UserForm1.Show
End If

End Sub
```

```
Private Sub ActivateFile_Click()
'***********************************************************************************************
'***********************************************************************************************
'*******                                                                              *********
'*******    This function turns on the program when the user clicks the activate file *********
'*******    button.                                                                   *********
'*******                                                                              *********
'***********************************************************************************************
'***********************************************************************************************
'*******  Created by Jeremy Lieberman   *******  December 13, 2011   *******   V1.0   **************
'***********************************************************************************************

Dim stcur As Worksheet

Set shtcur = ThisWorkbook.Sheets("Auto Home Page")

Call activatecode
shtcur.Protect ("Lieberman")
```

```
Unload Me
End Sub
```

```
Private Sub Clear_List_Click()
'*************************************************************************************************
'*************************************************************************************************
'*******                                                                                 *********
'*******    This function deletes all of the scheduled files currently in the automation  *********
'*******    file.                                                                        *********
'*******                                                                                 *********
'*************************************************************************************************
'*************************************************************************************************
'*******  Created by Jeremy Lieberman   *******  December 13, 2011   *******   V1.0   **************
'*************************************************************************************************

Dim lastcol As Integer, parse As Integer
Dim code As Integer

lastcol = Range("XFD2").End(xlToLeft).Column

'the next 12 rows delete the information stored for the user benefit
Columns(1).ClearContents
Columns(2).ClearContents
Columns(3).ClearContents
Columns(4).ClearContents
Columns(5).ClearContents
Columns(6).ClearContents
Columns(7).ClearContents
Columns(8).ClearContents
Columns(9).ClearContents
Columns(10).ClearContents
Columns(11).ClearContents
Columns(12).ClearContents

'the next 12 rows replaces the column titles.  Note that not all of these titles are visible
'to the user during operation of the file
Cells(1, 1) = "File to run"
Cells(1, 2) = "Time to Run"
Cells(1, 3) = "sequence to run"
Cells(1, 4) = "queries to run"
Cells(1, 5) = "user name"
Cells(1, 6) = "password"
Cells(1, 7) = "days to run"
Cells(1, 8) = "table name"
Cells(1, 9) = "excel to export"
Cells(1, 10) = "pdf to export"
Cells(1, 11) = "Reference Col"
Cells(1, 12) = "Last run"

'the following loop deletes the java code for all of the scheduled files
For parse = 14 To lastcol
    Columns(parse).ClearContents
Next

Unload Me
```

```
UserForm1.Show

End Sub
```

```
Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
'****************************************************************************************************
'****************************************************************************************************
'*******                                                                                   *********
'*******    This function activates if the user closes out of the userform.  It            *********
'*******    reactivates the files protection.                                              *********
'*******                                                                                   *********
'****************************************************************************************************
'****************************************************************************************************
'*******  Created by Jeremy Lieberman    *******  December 13, 2011    *******   V1.0   **************
'****************************************************************************************************

Dim stcur As Worksheet

Set shtcur = ThisWorkbook.Sheets("Auto Home Page")

shtcur.Protect ("Lieberman")

End Sub
```

D.3.2.2 Userform 2

This userform manages the query input for each file. Due to its smaller size, the code will not be separated into its separate functions but presented as a complete module.

```
Public filrow As Integer, i As Integer

'****************************************************************************************************
'****************************************************************************************************
'*******                                                                                   *********
'*******    This UserForm allows the user to input the name of each query within the       *********
'*******    database file along with the username and password, should they be needed      *********
'*******                                                                                   *********
'****************************************************************************************************
'****************************************************************************************************
'*******  Created by Jeremy Lieberman    *******  December 13, 2011    *******   V1.0   **************
'****************************************************************************************************

Private Sub Cancelbutton_Click()
'abort this userform
Unload Me
End Sub

Private Sub Okaybutton_Click()
'check that all necesary data has been entered and drop it into the excel file.  It recognizes
'if no username and password have been entered than it is a local query and those lines are not needed
```

```

'check that a query has been entered
If QueryName = "" Then
    MsgBox ("You have not enetered a query")
Else
    If Cells(filrow, 1) = "" Then Cells(filrow, 1) = "^ ^" 'drop a spacer into the name slot as new
lines are added
    Cells(filrow, 4) = UserForm2.QueryName                    'drop the query name into the excel file

    If UserName = "" Then                      'check if a username has been enetered for a query
        MsgBox ("The query has been added.  It is a local query and does not have a username and
password")
    Else
        Cells(filrow, 5) = UserForm2.UserName   'drops username into excel file
        Cells(filrow, 6) = UserForm2.Password   'drops password into excel file
        MsgBox ("The query has been added.")
    End If
    Unload Me
End If

End Sub

Private Sub Userform_activate()
'update the label so the user knows how many query's have been enetered so far
Label2.Caption = "Please enter the name of Query " & i & " that you want to process"
End Sub
```

D.3.2.3 Userform 3

This userform controls the input of date and time when each file will run. Due to its smaller size, the code will not be separated into its separate functions but presented as a complete module.

```
Public filrow As Integer

'*****************************************************************************************************
'*****************************************************************************************************
'*******                                                                                     *********
'*******    This UserForm allows the user to input both the database file and when that    *********
'*******    file will run (time and when during the week)                                   *********
'*******                                                                                     *********
'*****************************************************************************************************
'*****************************************************************************************************
'*******  Created by Jeremy Lieberman   *******  December 13, 2011   *******   V1.0   **************
'*****************************************************************************************************

'These functions rely on the property of prime numbers that if you multiply prime numbers together,
'the results will only be divisible by those prime numbers.  In this way, a unique number can be
'generated for each day combination that can easily be checked by seeing if the number is divisible by
'the respective day

```

```
Private Sub CommandButton1_Click()
'when the select file button is clicked, opens a dialog which will drop the selected file into box
TextBox2 = Application.GetOpenFilename("*.XXX, *.XXX")
End Sub

Private Sub Week_Days_2310_Click()
'****************************************************************************************************
'****************************************************************************************************
'*******                                                                                    *********
'*******    This sub is activated when the user presses the week days button and sets      *********
'*******    the buttons for each week day to turn on or off accordingly                    *********
'*******                                                                                    *********
'****************************************************************************************************
'****************************************************************************************************
'*******  Created by Jeremy Lieberman   *******  December 13, 2011   *******   V1.0   **************
'****************************************************************************************************

If Week_Days_2310 = True Then   'if the week day button is activated turn on each week day's button
    Monday_2 = True
    Tuesday_3 = True
    Wednesday_5 = True
    Thursday_7 = True
    Friday_11 = True
Else                            'if the week day button is deactivated turn off each week day's button
    Monday_2 = False
    Tuesday_3 = False
    Wednesday_5 = False
    Thursday_7 = False
    Friday_11 = False
End If
End Sub

Private Sub All_Week_510510_Click()
'****************************************************************************************************
'****************************************************************************************************
'*******                                                                                    *********
'*******    This sub is activated when the user presses the all week button and sets       *********
'*******    the buttons for each day to turn on or off accordingly                         *********
'*******                                                                                    *********
'****************************************************************************************************
'****************************************************************************************************
'*******  Created by Jeremy Lieberman   *******  December 13, 2011   *******   V1.0   **************
'****************************************************************************************************

If All_Week_510510 = True Then  'if the all week button is activated, turn on all other buttons
    Monday_2 = True
    Tuesday_3 = True
    Wednesday_5 = True
    Thursday_7 = True
    Friday_11 = True
    Saturday_13 = True
    Sunday_17 = True
Else                            'if the all week button is deactivated, turn off all other buttons
    Monday_2 = False
    Tuesday_3 = False
```

```vba
    Wednesday_5 = False
    Thursday_7 = False
    Friday_11 = False
    Saturday_13 = False
    Sunday_17 = False
End If
End Sub

Private Sub Continue_Click()
' finalize the data and drop it into the excel file

'the next if & else combo checks that the user inputs all of the information needed for the file
If TextBox2 = "" Then
    MsgBox ("You have not picked a file to process")
ElseIf Time = "" Then
    MsgBox ("You have not selected the time for the file to run")
ElseIf weekvalue = 1 Then
    MsgBox ("You have not selected a day for the file to run")
Else
    Cells(filrow, 1) = TextBox2.Value   'file name
    Cells(filrow, 2) = Time.Value       'time to run the file
    Cells(filrow, 7) = weekvalue        'unique value for the day combo, see below
    Unload Me
End If

End Sub

Private Sub Cancel_Click()
'abort the data entry
Unload Me
End Sub

Function weekvalue() As Double
'***************************************************************************************************
'***************************************************************************************************
'*******                                                                                   *********
'*******  This sub determines what the unique number is that represents this combination   *********
'*******  of days for use in the future to determine if the file should run on a given day *********
'*******                                                                                   *********
'***************************************************************************************************
'***************************************************************************************************
'*******  Created by Jeremy Lieberman   *******  December 13, 2011   *******   V1.0   **************
'***************************************************************************************************

Dim Date_Code As Double
Date_Code = 1

'the next section of code multiples the date code by the prime number for each day that was selected
If Monday_2 Then Date_Code = Date_Code * 2
If Tuesday_3 Then Date_Code = Date_Code * 3
If Wednesday_5 Then Date_Code = Date_Code * 5
If Thursday_7 Then Date_Code = Date_Code * 7
If Friday_11 Then Date_Code = Date_Code * 11
If Saturday_13 Then Date_Code = Date_Code * 13
If Sunday_17 Then Date_Code = Date_Code * 17
```

```
weekvalue = Date_Code        'returns the date code to the calling function

End Function
```

D.3.2.4 Userform 4

This userform manages the export of excel xls files. Due to its smaller size, the code will not be separated into its separate functions but presented as a complete module.

```
Public filrow As Integer    'global variable for what row has been selected

'*****************************************************************************************************
'*****************************************************************************************************
'*******                                                                                     *********
'*******    This UserForm allows the user to input export information so that database      *********
'*******    knows where and with what name each table should be exported                    *********
'*******                                                                                     *********
'*****************************************************************************************************
'*****************************************************************************************************
'*******  Created by Jeremy Lieberman    *******  December 13, 2011    *******   V1.0   ***************
'*****************************************************************************************************

'This userform is nearly identical to userfrom5 but for excel xls files instead of pdf files

Private Sub Cancel_Click()
'aborts the userform if the user presses cancel
Unload Me
End Sub

Private Sub Continue_Click()        'finalize the input data and drop it into the excel

Dim filepathlen As Integer

'this if statement checks to make sure that all information has been enetered
If table.Value = "" Then
    MsgBox ("You have not entered a table to export")
ElseIf export.Value = "" Then
    MsgBox ("You have not selected an export location")
Else
    Cells(filrow, 8) = table.Value      'update the table in the excel file

    If Include_Date = True Then
        filepathlen = Len(export.Value) - 4
        Cells(filrow, 9) = Mid(export.Value, 1, filepathlen) + "DATENOW.xls"  'add a note to replace
with the current date
    Else
        Cells(filrow, 9) = export.Value     'update the export location in the excel file
    End If
    Unload Me
```

```
End If

End Sub

Private Sub ExportFile_Click()
'provides input box for user to select the name and location of the export file
Dim temp As String
temp = Application.GetSaveAsFilename(table, "excel file (*.xls), *.xls", , "File to export the Table
to: If you pick an existing file it will be overwritten")
If (temp <> "False") Then export = temp
End Sub
```

D.3.2.5 Userform 5

This userform manages the export of pdf files. Due to its smaller size, the code will not be separated into its separate functions but presented as a complete module.

```
Public filrow As Integer    'global variable for what row has been selected

'***************************************************************************************************
'***************************************************************************************************
'*******                                                                                   *********
'*******    This UserForm allows the user to input export information so the database      *********
'*******    knows where and with what name each table should be exported                   *********
'*******                                                                                   *********
'***************************************************************************************************
'***************************************************************************************************
'******* Created by Jeremy Lieberman   ******* December 13, 2011   *******   V1.0   **************
'***************************************************************************************************

'This userform is nearly identical to userfrom4 but for PDF's instead of excel files

Private Sub Cancel_Click()
'aborts the userform if the user presses cancel
Unload Me
End Sub

Private Sub Continue_Click() 'finalize the input data and drop it into the excel

'this if statement checks to make sure that all information has been enetered
If table.Value = "" Then
    MsgBox ("You have not entered a table to export")
ElseIf export.Value = "" Then
    MsgBox ("You have not selected an export location")
Else
    Cells(filrow, 8) = table.Value      'update the table in the excel file
    If Include_Date = True Then
        filepathlen = Len(export.Value) - 4
        Cells(filrow, 10) = Mid(export.Value, 1, filepathlen) + "DATENOW.pdf"  'add a note to replace
with the current date
```

```
    Else
        Cells(filrow, 10) = export.Value      'update the export location in the excel file
    End If
    Unload Me
End If

End Sub

Private Sub ExportFile_Click()
'provides input box for user to select the name and location of the export file
Dim temp As String
temp = Application.GetSaveAsFilename(table, "pdf file (*.pdf), *.pdf", , "File to export the Table to:
If you pick an existing file it will be overwritten")
If (temp <> "False") Then export = temp
End Sub
```

D.3.3 Module Code

The code stored in modules is split into the following four sections:

- Run_Time_Code
- Run_Recognition
- Time_Order
- Other_Code

D.3.3.1 Module 1: Run_Time_Code

The code in Module 1 is split into the following functions with global variables included in the first section.

- Primary_Run
- Waitforfile
- RunNext
- Select_Next
- Weekdaycode
- GetText

```
'the next line allows the program to use the sleep function
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
'the next two variables are global variables used by the program while it is running
Dim nexttorun As Integer
Dim rowtorun As Integer

Sub Primary_Run()
'****************************************************************************************************
'****************************************************************************************************
'*******                                                                                    *********
'*******    This function schedules the run times for all of the programmed files           *********
'*******                                                                                    *********
'****************************************************************************************************
'****************************************************************************************************
'*******  Created by Jeremy Lieberman   *******  December 13, 2011   *******   V1.0   **************
'****************************************************************************************************

'******************************     Variable Declaration      ***************************************

Dim x As Integer, y As Integer       'variables for running through for loops
Dim totalruns As Integer             'stores the number of lines that the file will need to parse
Dim run_today As Integer             'local storage of which days to run the file
Dim shtcur As Worksheet              'stores the auto home page tab


'******************************     Initial Variable Set      ***************************************

Set shtcur = ThisWorkbook.Sheets("Auto Home Page")
run_today = weekdaycode      'pulls global variable data into function


totalruns = shtcur.Range("A1").End(xlDown).Row

rowtorun = 0    'presets global variable until this


'******************************     Primary Code            *******************************************

'The following for loop and series of if statements manage the sequencing for the files.
'Once the program is running routinely, this file is activated at midnight every day and sets
'all of the time stamps for every file.  When the program is started (presumably) during the
'middle of the day, it will only activate files that still need to run today
For x = 2 To totalruns   'This loop turns on the waits for every line of code
    If shtcur.Cells(x, 2) <> "" Then
        If Time < TimeValue(shtcur.Cells(x, 2).text) Then
            If shtcur.Cells(x, 7).Value Mod weekdaycode = 0 Then  'weekdaycode function below
                Application.OnTime TimeValue(shtcur.Cells(x, 2).text), "Waitforfile"
                If rowtorun = 0 Then                       'picks the first file entered
                    rowtorun = x                           'sets the rowtorun global variable
                    nexttorun = shtcur.Cells(x, 3)         'sets the nexttorun global variable
                ElseIf shtcur.Cells(x, 3) < shtcur.Cells(rowtorun, 3) Then
                                         ‘checks for the first scheduled file that still needs to be run
                    rowtorun = x                           'reset the rowtorun global variable
                    nexttorun = shtcur.Cells(x, 3)         'sets the nexttorun global variable
                End If
            End If
        End If
    End If
Next

Application.OnTime (#12:00:00 AM#), "Primary_Run"    'Programs the file to run the next day

End Sub
```

```
Sub Waitforfile()
'*****************************************************************************************************
'*****************************************************************************************************
'*******   This function puts a wait in place until the previous file has completed.  It   *********
'*******   works by creating a check for file which is deleted at the end of the run        *********
'*******   script.  The file has a 15 minute wait in case of a problem with the database   *********
'*****************************************************************************************************
'*****************************************************************************************************
'*******  Created by Jeremy Lieberman   *******  December 13, 2011   *******   V1.0   **************
'*****************************************************************************************************

'*******************************     Variable Declaration      ***************************************

Dim text As String
Dim shtcur As Worksheet

'*******************************     Initial Variable Set      ***************************************

Set shtcur = ThisWorkbook.Sheets("Auto Home Page")

'*******************************     Primary Code          ********************************************

shtcur.Unprotect ("Lieberman")                          'deactivate protection prior to running the file

If codeisactive Then                                    'global variable (on/off switch for the program)
    If (shtcur.Cells(rowtorun, 12).Value <> "") Then
        If (Date - shtcur.Cells(rowtorun, 12).Value) < 0.25 Then
                                                    'this prevents the program from running twice in a row
            GoTo 1
        Else
            shtcur.Cells(rowtorun, 12).Value = Date + Time
            Call RunNext                                       'function below
        End If
    Else
        shtcur.Cells(rowtorun, 12).Value = Date + Time
        Call RunNext                                           'function below
    End If
End If
1
shtcur.Protect ("Lieberman")                        'reactive the protection after the sub is complete
End Sub
```

```
Sub RunNext()
'****************************************************************************************************
'****************************************************************************************************
'*******                                                                                   *********
'*******   This function runs the database access file using the global variables to       *********
'*******   determine correct file to run                                                   *********
'*******                                                                                   *********
'****************************************************************************************************
'****************************************************************************************************
'******* Created by Jeremy Lieberman   ******* December 13, 2011   *******  V1.0   **************
'****************************************************************************************************

Dim shtcur As Worksheet
Dim script As String        'location where the database instructions will be saved
Dim databasepath As String      'the activation instructions for the database, linking to the txt file

Set shtcur = ThisWorkbook.Sheets("Auto Home Page")


script = CreateScript(rowtorun) 'calls sub to create instruction

databasepath = "\\path\file.exe " & splash & " -jscript " & Chr(34) & script & Chr(34)


Shell (databasepath)
Call Select_Next 'sets the program to run the next database file

End Sub
```

```
Sub Select_Next()
'****************************************************************************************************
'****************************************************************************************************
'*******                                                                                   *********
'*******   This function changes the file to run variables so that at the next time spot   *********
'*******   the correct database file will run                                              *********
'*******                                                                                   *********
'****************************************************************************************************
'****************************************************************************************************
'******* Created by Jeremy Lieberman   ******* December 13, 2011   *******  V1.0   **************
'****************************************************************************************************

'******************************     Variable Declaration      ***************************************

Dim x As Integer            'counting variable
Dim localnext As Integer    'stores a local copy of the next file that should be run
Dim shtcur As Worksheet     'stores the auto home page tab

'******************************     Initial Variable Set      ***************************************

Set shtcur = ThisWorkbook.Sheets("Auto Home Page")
localnext = nexttorun + 1    'adds one to the file tag to identify the next file that will run

'******************************     Primary Code        *******************************************

```

```
For x = 1 To 200                            'loop through the programming to find the next file
    If (localnext = shtcur.Cells(x, 3).Value) Then  'found the right file
        If (shtcur.Cells(x, 7).Value Mod weekdaycode = 0) Then

            nexttorun = localnext               'update the tag for the next file to run
            rowtorun = x                        'update the location tag for the next file to run
        Else
            localnext = localnext + 1
            x = 1
        End If
    End If

Next

End Sub
```

```
Function weekdaycode() As Integer
'**************************************************************************************************
'**************************************************************************************************
'*******  This function determines what day of the week it is and returns the appropriate  *********
'*******  prime number for the rest of the code                                              *********
'*******                                                                                     *********
'**************************************************************************************************
'**************************************************************************************************
'*******  Created by Jeremy Lieberman   *******  December 13, 2011   *******   V1.0   **************
'**************************************************************************************************

Dim datecode As Integer
datecode = Weekday(Date, vbMonday)

'The date code system relies upon the unique multiples of prime numbers.  Namely, a number generated
'by multiplying 2 prime numbers will only be divisable by those prime numbers.  In this way, a set of
'numbers which represents each possible day combination will exist that are cleanly divisable only by
'the associated prime numbers.  For example if the user wanted to have a file run on tuesday, wednesday
'and thursday.  The day code would be 3(tuesday) * 5(wednesday) * 7(thursday) or 105.  By checking if
'the date code divides cleanly into a given day's number (3 for tuesday) we can thus determine which
'days the user wanted the file to run in a single variable

If datecode = 1 Then        'Monday
    weekdaycode = 2
ElseIf datecode = 2 Then    'Tuesday
    weekdaycode = 3
ElseIf datecode = 3 Then    'Wednesday
    weekdaycode = 5
ElseIf datecode = 4 Then    'Thursday
    weekdaycode = 7
ElseIf datecode = 5 Then    'Friday
    weekdaycode = 11
ElseIf datecode = 6 Then    'Saturday
    weekdaycode = 13
ElseIf datecode = 7 Then    'Sunday
    weekdaycode = 17
End If
End Function
```

```
Function GetText(passFile As String) As String
'**************************************************************************************************
'**************************************************************************************************
'*******                                                                                  *********
'*******   This function pulls the text out of a txt file which is needed to determine    *********
'*******   if that file is empty (used to determine when the database has finished        *********
'*******   downloading                                                                    *********
'*******                                                                                  *********
'**************************************************************************************************
'**************************************************************************************************
'******* Created by Jeremy Lieberman   *******  December 13, 2011  *******   V1.0   **************
'**************************************************************************************************

   Dim File As Integer, Text As String

   ''Close any open text files
   Close

   ''Get the number of the next free text file
   File = FreeFile

   ''Write the entire file to sText, a local variable
   Open passFile For Input As #nSourceFile
   Text = Input$(LOF(1), 1)
   Close

   GetText = Text  'returns the full text of the txt file
End Function
```

D.3.3.2 Module 2: Run_Recognition

The code for module 2 is split into the following three functions. Again, global variables are included with the first function.

- Activatecode
- Deactivatecode
- codeonoff

```
Public codeisactive As Boolean  'global variable determining if the file is on or off
Public splash As String         'global variable determining if the program runs in the background

Sub activatecode()
'**************************************************************************************************
'**************************************************************************************************
'*******    This sub controls the activate codeisactive global variable which determines  *********
'*******    if the program is operating or not.  When it is false, the code is not       *********
```

```
'*******    running, when true it is running                                        *********
'******************************************************************************************
'******************************************************************************************
'*******  Created by Jeremy Lieberman  *******  December 13, 2011  *******  V1.0  **************

'*****************************     Variable Declaration     *************************************
Dim shtcur As Worksheet


Set shtcur = ThisWorkbook.Sheets("Auto Home Page")

shtcur.Unprotect ("Lieberman")
codeisactive = True 'turns on the global variable

'*****************************     Primary Code             *************************************

 'the with statement modifies the tracking button to indicate that the file is running
 With shtcur.Shapes("Rectangle 1")
    .Fill.ForeColor.RGB = RGB(78, 194, 64)
    .TextFrame.Characters.text = "The program is running"
    .TextFrame.Characters.Font.Color = RGB(255, 255, 255)
 End With

'the next part of the code sets the file sequence and triggers the timers
Call Time_Order      'determines sequence the files will run, located on the time_order module
Call Primary_Run     'runs the schedule function which will activate the code at the allotted time
Call deletescript    'removes any active script txts
Columns(12).ClearContents        'clears the recording of the last time each file was run
shtcur.Protect ("Lieberman")     'protects the file again
End Sub
```

```
Sub deactivatecode()
'******************************************************************************************
'******************************************************************************************
'*******    This code works like the activatecode sub but turns off the file so it won't  *********
'*******    keep running                                                                  *********
'*******                                                                                  *********
'******************************************************************************************
'******************************************************************************************
'*******  Created by Jeremy Lieberman  *******  December 13, 2011  *******  V1.0  **************
'******************************************************************************************

Dim shtcur As Worksheet

Set shtcur = ThisWorkbook.Sheets("Auto Home Page")

shtcur.Unprotect ("Lieberman")

codeisactive = False    'turns off the global variable

 'the next with statement updates the button that indicates to the user if the file is on
 With shtcur.Shapes("Rectangle 1")
    .Fill.ForeColor.RGB = RGB(226, 35, 30)
    .TextFrame.Characters.text = "The program is off"
    .TextFrame.Characters.Font.Color = RGB(0, 0, 0)
 End With

shtcur.Protect ("Lieberman")    'reprotect the sheet
End Sub
```

```
Sub codeonoff()
'*******************************************************************************************
'*******************************************************************************************
'*******                                                                           *********
'*******    This sub flips the activate state, (on when the file is off and vice versa)    *********
'*******                                                                           *********
'*******************************************************************************************
'*******************************************************************************************
'*******  Created by Jeremy Lieberman   *******  December 13, 2011   *******   V1.0   ***************
'*******************************************************************************************

If codeisactive = False Then
Call activatecode                   'turn on
Else
Call deactivatecode                 'turn off
End If

End Sub
```

D.3.3.3 Module 3: Time_Order

Module 3 only contains one function, time_order, for sorting the order that the files will run based upon their scheduled times.

```
Sub Time_Order()
'*******************************************************************************************
'*******************************************************************************************
'*******                                                                           *********
'*******    This function sets the order that the queries will be run              *********
'*******                                                                           *********
'*******************************************************************************************
'*******************************************************************************************
'*******  Created by Jeremy Lieberman   *******  December 13, 2011   *******   V1.0   ***************
'*******************************************************************************************

'****************************     Variable Declaration     ***********************************

Dim totalruns As Integer            'stores the number of files that have entered
Dim lastrun As Integer              'the last row which has a file to run
Dim Timestorun(1 To 200)            'Array containing up to 200 files to store and run
Dim x As Integer, y As Integer      'temp variables for sorting through the data
Dim A, AA '                         'variables for transferring data to the array
Dim shtcur As Worksheet             'auto home page tab

'****************************     Initial Variable Set     ***********************************

```

```
Set shtcur = ThisWorkbook.Sheets("Auto Home Page")

y = 1 'set the default start value of y for parsing the array

lastrun = shtcur.Range("A65536").End(xlUp).Row
totalruns = 0                                     'initializes the function to 0 files to run

'*******************************     Primary Code          **************************************

'the for loop parses through the stored data to determine how many files to run as well as the order
For x = 2 To lastrun
    If (shtcur.Cells(x, 2).text <> "") And (shtcur.Cells(x, 2).text <> "^ ^") Then 'conditions to
ignore
        A = TimeValue((shtcur.Cells(x, 2).text))
        totalruns = totalruns + 1
        For y = 1 To 200
            If (A < Timestorun(y)) Then ' if the current time is sooner it puts it earlier in the array
                AA = Timestorun(y)
                Timestorun(y) = A
                A = AA
            ElseIf (Timestorun(y) = "") Then ' at the end of the array add the value and end the loop
                AA = Timestorun(y)
                Timestorun(y) = A
                A = AA
                Exit For
            End If
        Next
    End If
Next

'this for loop places the run sequence into the excel file for future use
For x = 2 To lastrun
    For y = 1 To 200
        If (shtcur.Cells(x, 2).text) <> "" Then
            If TimeValue((shtcur.Cells(x, 2).text)) = Timestorun(y) Then
                shtcur.Cells(x, 3) = y
                Exit For
            End If
        End If
    Next
Next

End Sub
```

D.3.3.4 Module 4: Other_Code

```
Sub ClearScript()
'****************************************************************************************************
'****************************************************************************************************
'*******                                                                                    *********
'*******    This sub clears the script txt file, leaving an empty file                      *********
'*******                                                                                    *********
```

```
'**************************************************************************************************
'**************************************************************************************************
'*******  Created by Jeremy Lieberman   *******  December 13, 2011   *******   V1.0   **************
'**************************************************************************************************

sFile = "C:\Path\Script.txt"
iFileNum = FreeFile
Open sFile For Output As iFileNum
Close #iFileNum
End Sub
```

```
Function CreateScript(ByVal rowtorun) As String
'**************************************************************************************************
'**************************************************************************************************
'*******                                                                                  *********
'*******    This sub creates the script txt that will contain the database’ s code        *********
'*******                                                                                  *********
'**************************************************************************************************
'**************************************************************************************************
'*******  Created by Jeremy Lieberman   *******  December 13, 2011   *******   V1.0   **************
'**************************************************************************************************

'******************************     Variable Declaration     *************************************
Dim script As String        'stores the location where the script txt will be saved on the hard drive
Dim fnum As Integer         'paramater for accessing the txt file
Dim Script_Num As Integer   'gives the script txt a unique name depending on its sequence number
Dim filename As String      'the name of the txt file
Dim shtcur As Worksheet     'stores the auto home page
Dim Path_With_Date As String
'******************************     Initial Variable Set     *************************************

Set shtcur = ThisWorkbook.Sheets("Auto Home Page")

Script_Num = shtcur.Cells(rowtorun, 3)              'pulls the sequence number to add to the file name
filename = "script" + CStr(Script_Num) + ".txt"

script = "C:\\Path\" + filename            'file location
fnum = FreeFile                                     'gives fnum a value for creating the file

'******************************     Primary Code             *************************************

Open script For Output As fnum          'open up the txt file to write the instructions
colnum = shtcur.Cells(rowtorun, 11)
x = 2
Do While shtcur.Cells(x, colnum).text <> ""     'writes the database instructions to the txt

'the following line replaces the "DATENOW" code with the actual date
Path_With_Date = Replace(shtcur.Cells(x, colnum).text, "DATENOW", Replace(Date, "/", "-"))

Print #fnum, Path_With_Date


x = x + 1
Loop
Close #fnum
```

```

CreateScript = script             'has the function return the name of the file for use in calling
function
End Function
```

```
Sub getdata()
'****************************************************************************************************
'****************************************************************************************************
'*******                                                                                   *********
'*******    This function activates the dialog for entering new files                      *********
'*******                                                                                   *********
'****************************************************************************************************
'****************************************************************************************************
'*******  Created by Jeremy Lieberman   *******  December 13, 2011   *******   V1.0   **************
'****************************************************************************************************

UserForm1.Show       'opens the first userform
End Sub
```

```
Function cutpath(ByVal path As String) As String
'****************************************************************************************************
'****************************************************************************************************
'*******                                                                                   *********
'*******    This function returns the file name from the file path                         *********
'*******                                                                                   *********
'****************************************************************************************************
'****************************************************************************************************
'*******  Created by Jeremy Lieberman   *******  December 13, 2011   *******   V1.0   **************
'****************************************************************************************************

'*******************************     Variable Declaration      **************************************

Dim parse As Integer          'looks through the file path in reverse until it finds a  "\"
Dim begin_found As Boolean    'determines when the file has selected the file name from the path

'*******************************     Initial Variable Set      **************************************

begin_found = False           'starts with not having the file name
parse = Len(path)             'path starts at the last letter of the string

'*******************************     Primary Code              **************************************

Do While parse > 1 And begin_found = False  'ends when parse reaches the end of the name or the file
name is found

If (Mid(path, parse, 1) = "\") Then 'file name is everything after the last \ mark
    begin_found = True  'sets the file to having found the full file name
End If

parse = parse - 1         'continue parsing backwards through the file path
Loop

cutpath = Mid(path, parse + 2)      'the sub returns teh file name

```

```
End Function
```

```
Sub deletescript()
'***************************************************************************************************
'***************************************************************************************************
'*******                                                                                   *********
'*******    Deletes the script txt if it becomes necessary                                 *********
'*******                                                                                   *********
'***************************************************************************************************
'***************************************************************************************************
'*******  Created by Jeremy Lieberman   *******  December 13, 2011   *******   V1.0   **************
'***************************************************************************************************

If Dir("C:\path\script.txt") = "Script.txt" Or Dir("C:\path\script.txt") = "script.txt" Then
    Call ClearScript        'wipe the script file if it exists
End If
End Sub
```

```
Sub nosplash()
'***************************************************************************************************
'***************************************************************************************************
'*******                                                                                   *********
'*******    Changes the "splash" condition.  When the splash variable is turned off        *********
'*******    the program will run in the background when processing each script             *********
'*******                                                                                   *********
'***************************************************************************************************
'***************************************************************************************************
'*******  Created by Jeremy Lieberman   *******  December 13, 2011   *******   V1.0   **************
'***************************************************************************************************

'******************************     Variable Declaration      **************************************

Dim shtcur As Worksheet


Set shtcur = ThisWorkbook.Sheets("Auto Home Page")
shtcur.Unprotect ("Lieberman")

If splash = "-nosplash" Then            'if global variable is on: turns it off
    splash = ""                         'resets the global variable to blank
    With shtcur.Shapes("Button 2")
        .TextFrame.Characters.text = "Hide until run is complete"   'updates the splash button
    End With
Else
    splash = "-nosplash"                'resets the global variable to no splash
    With shtcur.Shapes("Button 2")
        .TextFrame.Characters.text = "Show during run" 'updates the splash button
    End With

End If

shtcur.Protect ("Lieberman")
End Sub
```