Quantitative Selection of Inspection Plans for Variation Risk Management

by

Tony J. Chen

B.S. Mechanical Engineering
University of California at Berkeley, 1997

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN MECHANICAL ENGINEERING
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE 1999

Signature of Author.....
Department of Mechanical Engineering
May 7, 1999

Certified by.....
Anna Thornton
Assistant Professor of Mechanical Engineering
Thesis Supervisor

Accepted by.....
Ain A. Sonin
Professor of Mechanical Engineering
Chairman, Committee for Graduate Students

Quantitative Selection of Inspection Plans for Variation Risk Management

By

Tony J. Chen

Submitted to the Department of Mechanical Engineering
on May 7, 1999, in Partial Fulfillment of the
Requirements for the Degree of Master of Science in
Mechanical Engineering

ABSTRACT

Over the last decade, the importance of quality has increased significantly. Quality improvement efforts involve mitigating the impact of manufacturing variation through robust design, statistical process control (SPC), and inspection. This thesis focuses on the last: how to choose an inspection plan to remove the most variation at the lowest cost. The optimal inspection plan balances the cost of inspection and rework against the cost of increased quality. This thesis describes an empirical analysis and prototype software that employs Monte Carlo simulation and simulated annealing to identify the optimal inspection plan. This thesis demonstrates the functionality of the theory and the software through an example from the aircraft industry.

Thesis Supervisor: Anna Thornton

Title: Assistant Professor of Mechanical Engineering

ACKNOWLEDGEMENT

For me, this research has been a rich and rewarding experience. I owe a debt of gratitude to all those who have given me guidance and support.

I would like to thank Professor Anna Thornton for being such an inspiring research advisor. Her zeal for excellence has been a major driving force behind this thesis.

Thanks to the Center for Innovation in Product Development for funding this research. I would also like to recognize Chuck Hura and Sam DeLuca from industry for being the sponsors and mentors of my summer internship. Everyone else at the company had also enriched my experience in some way.

Special thanks to my family and all my friends for their support and love throughout the years. Thanks to Derrick and Michael for being such inspirational models, showing me that completing a thesis is not an impossible feat. I am also grateful for Anita for giving me the encouragement when I needed them and the discipline to keep writing when I had none. Also, thanks to Julia, Loo, Esa, and Joseph, for being the family outside of family. Along with Jeff, Wendy, Ernie, Connie, Tom, Linda, Tina, Sonja, and other friends at Boston, Berkeley, and Rochester, they have helped bringing excitement and sanity into my otherwise monotonous life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Companies cannot survive without providing high quality products. To produce high quality products, the design and manufacturing community are using a variety of tools to improve quality throughout the product development cycle: e.g., Six Sigma [Clifford 1997; Doherty 1997], process improvement, inspection, statistical process control (SPC), process change, and robust design. Because no single approach is superior in every case, product development team often utilizes a combination of approaches to achieve the highest quality at the lowest cost [Thornton 1998]. To evaluate the different approaches, companies need to understand the cause-and-effect relationship between the feature-level variation and the system-level product quality. In addition, they must be able to model the quality costs and the impact of different variation reduction strategies.

One of these strategies, inspection, is the focus of this thesis. Although significant work has been done to determine the optimal inspection limits for a single feature, no work to date has developed a systematic method for identifying where in the assembly of a product inspection should occur. On one hand, inspecting, scrapping, and reworking parts at the part feature level is relatively inexpensive; however, the ability to influence the over-all quality of the product is small. On the other hand, inspecting, reworking, and scrapping at the system level is very expensive; however, the ability to directly influence the final product quality is higher.

This thesis describes an empirical analysis and user-friendly software employing Monte Carlo simulation and simulated annealing to identify the optimal inspection plan for a complex product. This thesis demonstrates the functionality of the theory and the software through an example from the aircraft industry. Section 1.1 describes a method to identify where excess variation significantly affects product quality, Section 1.2 describes other research in the area of inspection, Section 1.3 defines the problem statement, and Section 1.4 describes the approach of this research.

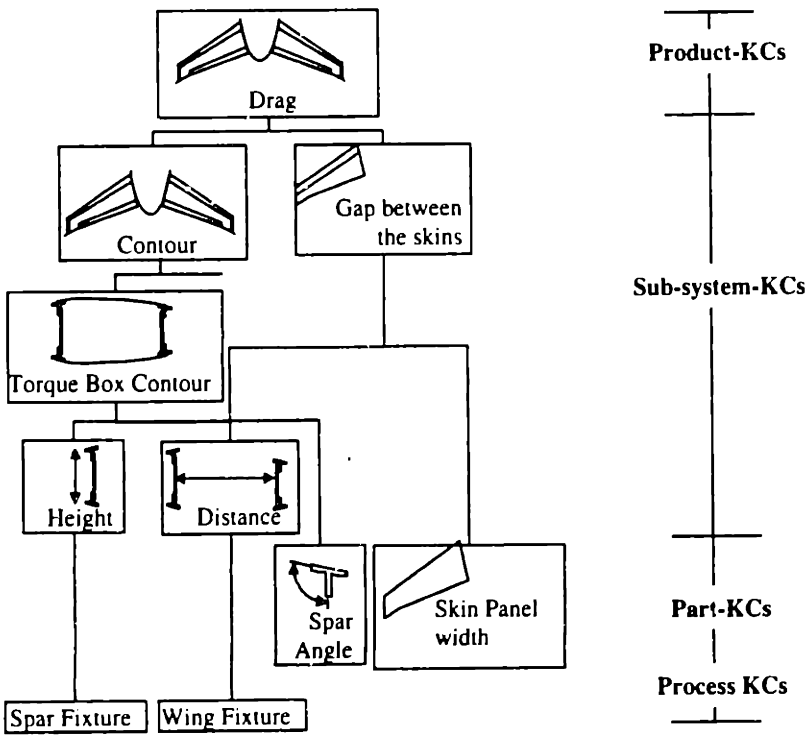
1.1 Key Characteristics

In a complex product, it is not economically or logistically feasible to control and/or monitor the thousands of tolerances specified in a drawing set. To identify what tolerances to control, many organizations are using methods called Key Characteristics (KCs), which are also termed Critical Parameters, Critical-to-Function Responses, Critical-to-Quality Variables, and Special Characteristics, by

different companies[1]. KC methods are used to identify and communicate to the entire design team where excess variation will most significantly affect product quality [Lee and Thornton 1996].

First, design teams identify where system-level variation may impact performance, safety, and fit requirements. Performance requirements are determined by the market and customers, safety requirements are set by internal and external (i.e., FAA) safety groups, and fit requirements are issues that can make manufacture and assembly difficult (i.e., loading a part into a fixture). The variation-sensitive product quality requirements are called *product-KCs*. Second, the design team flows these requirements through the product decomposition. The source of variation for the Product-KCs can often be contributed to variations occurring throughout the manufacturing and assembly processes.

Figure 1 shows a KC flowdown for the aircraft wing drag. The product-KC (drag) at the top of the tree has several contributing subsystem-KCs (contour and gap). The subsystem-KCs are, in turn, a function of the part-KCs (spar angle) and process-KCs (fixtures). In some cases, it may be possible to flow the KCs down to the process level; however, there may be cases where parts are delivered by outside suppliers. In this case, part-KCs become product-KCs for the supplier.



[1] For consistency, the thesis will use the term Key Characteristic (KC).

Figure 1: KC Flowdown

A flowdown typically has multipie decomposition levels. In summary, the KC flowdown provides a system view of potential variation risk factors and captures the design team's collective knowledge about variation, its contributors, and the way variation propagate from the feature level to the system level. Traceability between process- and product-KCs through part- and subsystem-KCs is necessary to identify what process should be controlled, to identify where process change may positively impact product quality, and to identify root causes of quality problems. Putting resources in a prioritized list of KCs seeks to achieve high product quality in the most cost-efficient way.

1.2 Research on Inspection

While variation risk mitigation can be achieved by different options — design changes, process changes, process improvement, and inspection — this thesis focuses on inspection. Early papers in the field focused primarily on the effect of inspection error on quality control [Mei, Case et al. 1975; Menzefricke 1984]. The two types of error they considered were type-I errors (a conforming item was rejected) and type-II errors (a nonconforming item is accepted).

To optimize expected profit of the production process, the *quality selection* (also termed *economic selection*) problem was also widely studied. Quality selection maximizes profits by optimizing the quality levels. Hunter and Kartha [1977] and Bisgaard *et al.* [1984] formulated the problem of selecting a process mean to maximize the expected profit. Carlsson [1989] considered lot rejection based on the average amount of variability in the sample containers. To include the economic effect of inspection error on the economic selection problem, Chen and Chung [1996] proposed a method to select the optimal inspection precision level based on the number of repeated readings on a gauge and the error factors of bias and imprecision. While the above researchers demonstrated that inspection can be adjusted to maximize expected profit, they do not directly address selecting an inspection plan for an assembled product.

Some researchers address strategic allocation of inspection in multi-product production systems. By treating inspection as stations in the system, some studies have focused on serial [Chakravarty and Shtub 1987] while others focused on non-serial [Elmaghraby 1986] production process. Viswanadham *et al.* [1996] presented stochastic search techniques for locating inspection stations in a multistage manufacturing system. However, they assumed a given, independent, acceptance probability for the entire product at each stage of assembly. In addition, their models focus on inspecting out bad products rather than inspecting out parts that will create bad products.

Some researchers have addressed the multi-characteristic problem, Greenshtein and Rabinowitz [1997] developed methods to allocate inspection for multi-characteristic items. Rabinowitz and Emmons [1997] then presented guidelines for scheduling multiple inspection tasks for a single inspection facility. However, both papers only considered a serial two-stage inspection program. In addition, they only proposed heuristics for cases with more than two stages of production.

In summary, a variety of mathematical models have been proposed that set the optimal inspection by minimizing an objective cost function. While their mathematical models provide insight, there is no work that optimizes an inspection plan for an assembly with multi-characteristic specifications. In addition, all research up-to-date have used an independent constant acceptance probability for inspection. How inspection changes the distribution was not considered.

1.3 Problem Statement

Industry faces a lack of quantitative method to help chose the best variation reduction strategy. Since variation reduction efforts, such as inspection, depend on limited resource, manufacturers need to strategically allocate their resources. They look for answers to the following three questions relating to inspection:

What features should be inspected? Out of the hundreds or thousands of dimension requirements on the drawing, only a few features make significant impact on variation related quality issues. Those high impact KCs need to be identified so quality improvement resources can be focused on them.

Variation sensitive quality issues can also arise from three different causes. First, the product features may have tight tolerance requirements. This is caused by high performance, safety, or fit requirements. Second, the contributing parts or processes may have high variation to start with. Often, process improvement or process change has minimal effect, or too costly, to reduce variation. Or third, slight variation in the contributing parts or processes results in high product-KC variation. This also indicates a design not robust to variation. Resources can be applied to minimize or eliminate the three causes.

In addition, designing and manufacturing teams often ask the question: *where in the assembly process should we focus our inspection resources.* When features are inspected before parts are assembled, the cost of scrap or rework is low, but the ability to directly impact final product quality is also low. However, if features are inspected after parts are assembled, the cost of scrap or rework is high, but the ability to directly impact final product quality is also high. A method to perform cost-benefit tradeoff between different inspection plans is needed.

What are the inspection tolerances? Deciding where in the product architecture to place the inspection checkpoints is not the only decision engineers have to make; they must also set the inspection limits. Imposing stringent inspection requirements may result in unnecessary scrap or rework. Scrap and rewor have different costs and different impacts. The opposite may result in product-KCs not meeting specification. An optimal inspection tolerance balances the two extremes.

Which corrective action to use? After inspection identifies nonconforming units, engineers need to decide whether to rework the features or scrap the entire part or subassembly. The decision should be based on a quantitative cost and benefit tradeoff for the entire production system.

The goal of this research is to develop a methodology so companies can use the Key Characteristics approach to reduce the risk of variation. Often time, companies use a built-and-test approach when dealing with quality issues. However, a set of analysis tools for Key characteristics will enable engineers to analysis the cost and benefits of different variation risk management approaches, including inspection, earlier in the design process. By dealing with the issue of variation early in the design process, engineers can save valuable prototyping and ramp-up time. This can help reduce product cycle time and results in higher quality products at lower cost.

1.4 Research Approach

This research uses two approaches to analysis the optimal inspection problem. First, mathematical equations are used to model and analysis the cost and impact of inspection. Second, a simulation based optimization approach is used to achieve the same goal.

The analytical equations are first formulated from a simple variation stack-up model where one out of four parts are inspected before assembly. Instead of treating the acceptance probabilities as independent constant probabilities, they are calculated explicitly based on the process capabilities and the inspection tolerances. The equations are then extended into generalized equation that can be applied to the general case where multiple inspected and uninspected parts are assembled with non-unity variation propagation relations. Another set of equations is also formulated to deal with the KC flowdown model where variation propagation through multiple layers, as described in Section 1.1.

To achieve the goal of analyzing and selecting inspection plans for complex products, the simulation-based approach is done in three steps as shown in Figure 2.

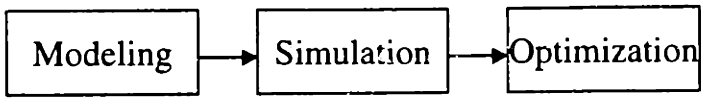


Figure 2: The Three-Step Approach to Quantitatively Select an Optimal Inspection Plan

Modeling. Just like the analytical approach, the modeling of inspection for the simulation-based approach is also based on the framework of the KC flowdown with an additional product assembly hierarchy. The general model is a non-serial multistage assembly process building a multi-characteristic product.

Simulation. To captures the effect of variation propagation, simulation uses a Monte Carlo simulation.[2] In addition, the simulation models the effect of different corrective action strategies such as scrap and rework.

Optimization. Once we can use the simulation to model variation propagation and the effect of inspection and corrective actions, a stochastic optimization algorithm is then used to answer the question: *what inspection plan achieves the highest quality improvement at the lowest cost.* A prototype software, *KCTool*, is used as the modeling backbone for this research [Ardayfio 1998; Thornton 1999]. *KCTool* enables a designer to capture the product architecture and KC flowdown. In addition, the software provides visual feedback about the levels of risk as well as the optimal locations for inspection.

The problem is first described in Chapter 2 using the math-based approach, which derives the optimal inspection equations for analyzing inspection problems from simple variation stack-up problems then expending it to general multi-layer, multiple inspection analysis. Chapter 3 describes the inspection modeling process used in the *KCTool* environment. Chapter 4 describes the inspection simulation process. Chapter 5 describes the inspection optimization process and an example. Chapter 6 contains the conclusions.

[2] Even though the distributions are assumed to be gaussian, root sum squared (RSS) approximations can not be used to predict the final product quality. Inspection and rework truncate the distributions and invalidate the RSS calculations.

NOTATION

x_i	the value of feature i in the four-part assembly;	n	Number of parts in an assembly, or number of KCs
x_{2-4}	the combined value of the other three features in the four-part assembly;	$C_{i,k}$	the cost of inspecting feature k
y	the value of the product feature	C_f	the cost of not meeting the system specification limits
σ_i	process standard deviation for feature i;	$C_{p,k}$	the cost of scrapping item k
$[LL_{sys}, UL_{sys}]$	The lower and upper specification limit for the entire system	C_r	the cost of reworking a feature
$[LL_i, UL_i]$	The lower and upper specification limit for part or KC i	C_m	the original product cost
m	nominal specification	C_T	the total expected cost of the inspection plan
μ	the mean of process capability	$C_{inspection}$	the total inspection cost of producing one product
t_i	number of standard deviation away from nominal for feature i	C_{scrap}	the total scrap cost of producing one product
p_1	*the probability of accepting part 1 during inspection*	C_{rework}	the total rework cost of producing one product

P_y	the probability of finding y acceptable	$C_{failure}$	the total failure cost of one product
$p_{y\|x_i}(x_i)$	the probability of finding y within-spec for a given value of x_i	P_{accept}	the probability of accepting the change for simulated annealing
$pdf(x_i)$	the probability density function of x_i	T	The temeprature of the state for simulated annealing
$pdf'(x_i)$	the probability density function of x_i after inspection and scrapping	T_{start}	The starting temperature for simulated annealing
$pmf(x_i)$	the probability mass function of x_i	T_{end}	The ending temperature for simulated annealing
c_i	the transfer coefficients of x_i to the next level	α	The ratio for simulated annealing temperature decrease, $\alpha<1$
Vector S	Keep track of whether or not KCs are inspected	*Vector A*	Keep track of whether to scrap the part or rework the out-of-spec KCs
Vector T	Keep track of the inspection tolerance settings		

CHAPTER 2: OPTIMUM INSPECTION ANALYSIS

2.1 Introduction

This section derives the mathematical frame works for solving an optimal inspection problem where multiple sources of variation propagate through many intermediate layers and ultimately contribute to the final variation at the customer requirement level. Due to the complexity of the issue, the mathematics formulation is divided into three sections. First, the analysis starts with a simple four-block variation stack-up problem and calculates the total cost associated with inspecting one out-of-four source of variation. Second, the analysis is extended to multiple inspections and eventually forming a generalized equation that can be used to model the cost and benefit of inspecting k out-of-n sources. Third, the analysis calculates a new probability density function (pdf) based on pdfs of the variation sources, both inspected and uninspected.

2.2 Single Inspection

We will start our analysis with a simple four-part variation stack-up problem shown in Figure 3. The purpose of this analysis is to determine the optimal inspection limit for one of the four blocks. While the assembly stack-up architecture is simple, it provides a foundation for extending the optimal inspection limit analysis to more complex product architectures. As in Equation 1, the product-KC, y, is a function of the four block lengths, x_i, and the variation of the four parts contribute to a variation of y.

$$y = x_1 + x_2 + x_3 + x_4 \tag{1}$$

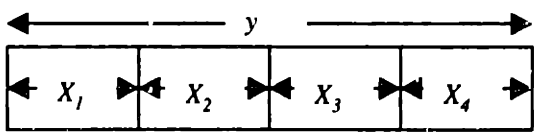


Figure 3: The Assembly of Four Parts

We assume each part has a Gaussian (or normal) distribution with standard deviation σ_i, and the probability density function of a Gaussian distribution is defined as

$$pdf(x_i) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{\frac{(x_i-\mu_i)^2}{2\sigma_i^2}} \tag{2}$$

When no part is inspected, we can calculate the probability density function for y using the root-sum-square method. So,

$$pdf(y) = \frac{1}{\sqrt{2\pi}\sigma_y} e^{\frac{(y-\mu_y)^2}{2\sigma_y^2}} \tag{3}$$

where

$$\sigma_y = \sqrt{\sigma_1^2 + \sigma_2^2 + \sigma_3^2 + \sigma_4^2} \tag{4}$$

For the analysis, we are assuming part 1 will be inspected and the others not. To simplify the calculation, we combine the three uninspected parts, x_2, x_3, and x_4, into a single variable, $x_{2\text{-}4}$, which has a Gaussian distribution, $pdf(x_{2\text{-}4})$, with a standard deviation of

$$\sigma_{2-4} = \sqrt{\sigma_2^2 + \sigma_3^2 + \sigma_4^2} \tag{5}$$

When part 1 is inspected, if the value falls outside the inspection limits, UL_1 and LL_1, the part is scrapped. The inspection upper limit, UL_1, and lower limit, LL_1, can be expressed by Equation 3 and 4, where a scalar t_1 represents the tightness of tolerance allocation[3] and m_1 is the desired nominal.

$$UL_1 = m_1 + t_1\sigma_1 \tag{6}$$

$$LL_1 = m_1 - t_1\sigma_1 \tag{7}$$

We can construct the decision tree to account for all scenarios when part 1 is inspected. Figure 4 shows the decision tree, the probabilities of each outcome, and the costs associated with the outcomes. C_f is the cost of not achieving a system target, C_p is the cost of scrapping a part, and C_i is the cost of inspecting a part. C_m is the cost of manufacturing the unit without the inspection cost. The original manufacturing cost is the same for all branches, so it can be added to the total cost separately.

[3] The value t_1 enables the analysis to be normalized.

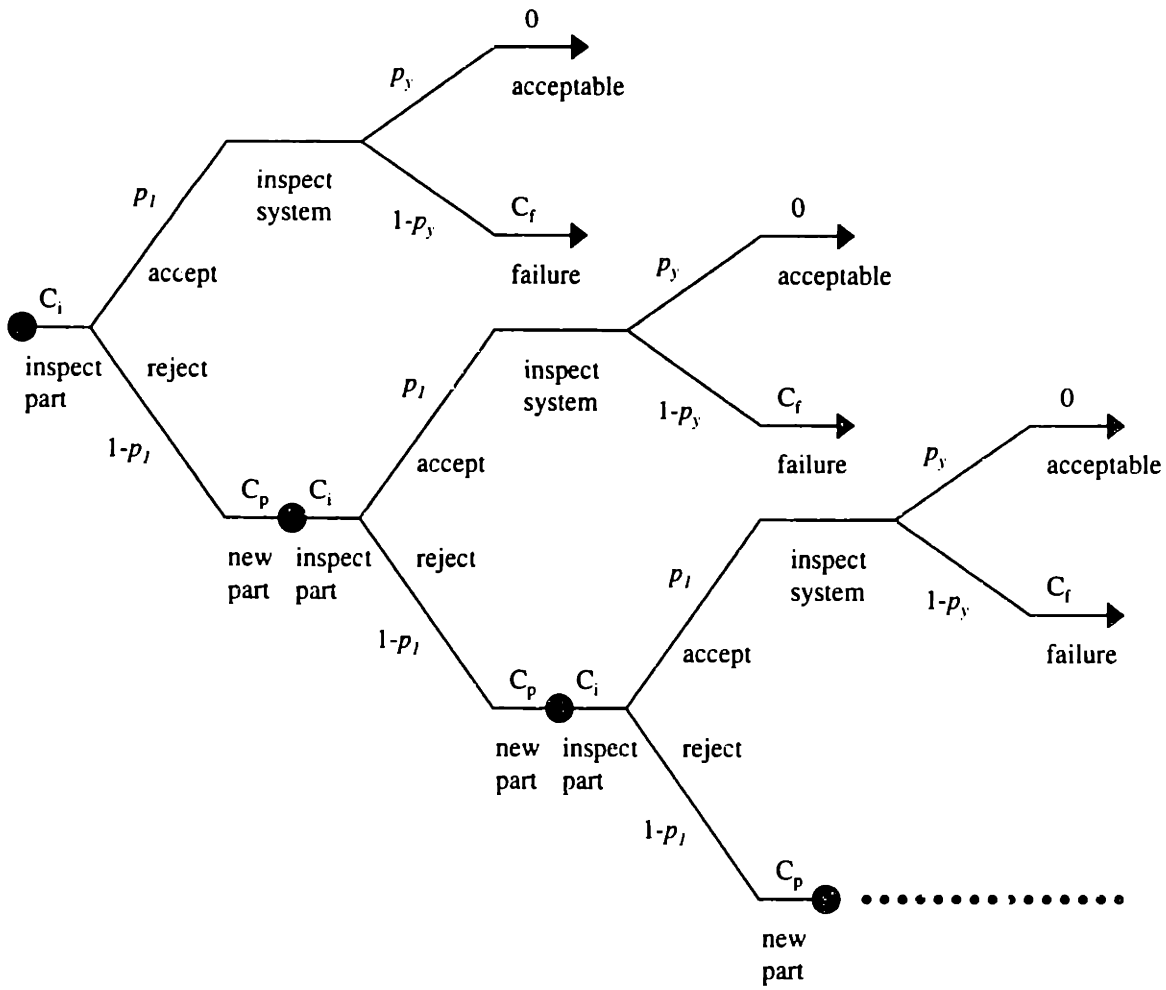


Figure 4: Probability Tree Diagram for Single Inspection

We start the process by inspecting the part before the assembly process. If the part falls within inspection limits, a probability of p_I, it is accepted and the system is assembled. Then the system is inspected within specification. The probability of accepting the system is p_y. If the part is rejected, a new part is picked from the lot and inspected. The outcomes of the second inspection are identical to the first. As a result, an infinite decision tree captures all possibilities.

2.2.1 Cost Calculation

We can calculate the total expected cost by just focusing on the repetitive part of the probability tree. Figure 5 shows the recursive inspection routine. We can also represent the rejection branch after a new part replaces the scrapped part as a module identical to the whole probability tree.

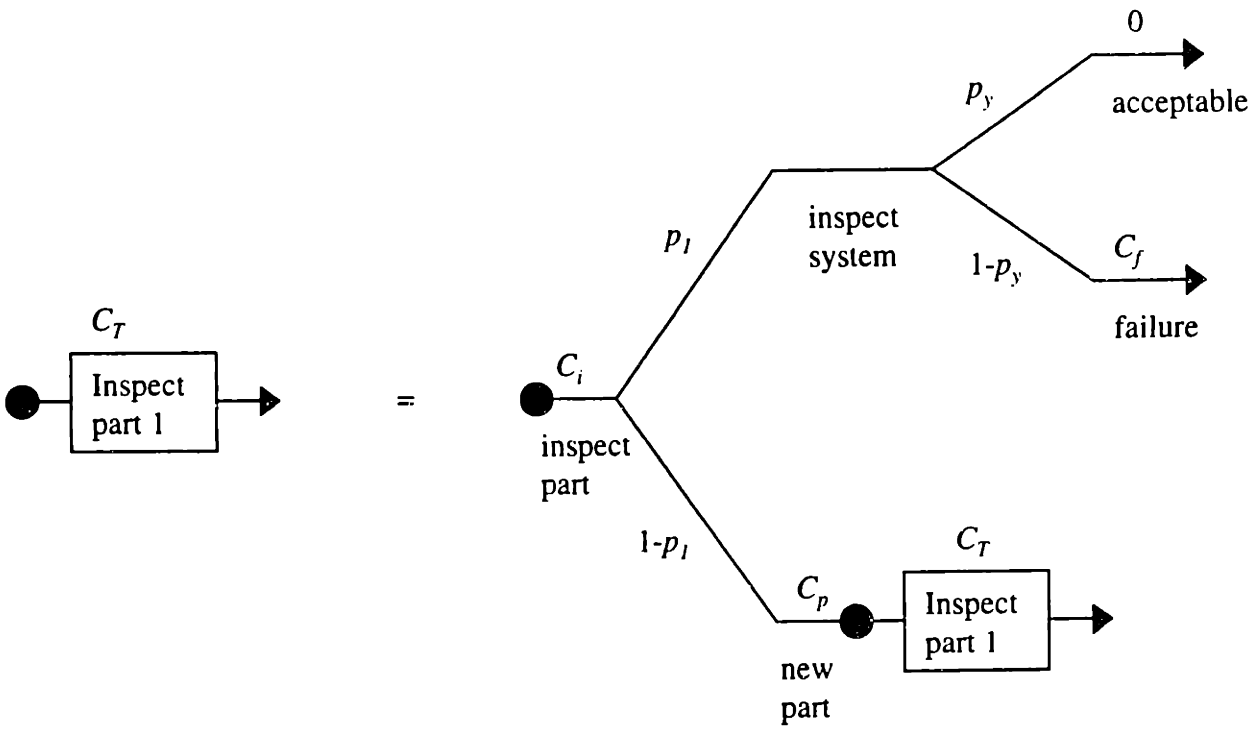


Figure 5: Representing the Decision Tree as Modules

The total expected cost of the whole module, C_T, can be calculated by summing the cost of each branch multiplied by the probability of the branch

$$C_T = C_{i_1} + p_1 \cdot (1 - p_y) \cdot C_f + (1 - p_1) \cdot (C_{p_1} + C_T) \tag{8}$$

We can express a final equation for C_T as

$$C_T = \frac{1}{p_1} \cdot C_{i_1} + (1 - p_y) \cdot C_f + \frac{1 - p_1}{p_1} \cdot C_{p_1} \tag{9}$$

2.2.2 Probability Calculation

Two probabilities, p_1 and p_y, need to be calculated because they are functions of the inspection tolerance tightness, t.

The probability of accepting part 1 when it is inspected, p_1:

$$p_1 = \int_{LL_1}^{UL_1} pdf(x_1) dx_1 \tag{10}$$

where $pdf(x_1)$ is the Gaussian distribution for part 1 given a standard deviation of σ_1 and a mean shift of μ_1-m_1.

Converting the equation to a function of part 1's inspection tolerance setting, t_1, results in:

$$p_1(t_1) = \int_{LL_1(t_1)}^{UL_1(t_1)} pdf(x_1)\,dx_1 \tag{11}$$

When the inspected dimension of x_1 is assembled with the rest of the system, the probability of accepting the product is a function of x_1, and the amount of variation, $\sigma_{2\text{-}4}$, in other parts. For each value of x_1, the probability of accepting the system is a conditional probability, $p_{y|x1}$ (this is interpreted as "the probability of finding the system within specification if the part has variation of x_1). Using the joint probability distribution equation [Drake 1967], the expected probability that the product is acceptable, p_y can be calculated as:

$$p_y = \int_{-\infty}^{\infty} pdf(x_1)\cdot p_{y|x_1}(x_1)\cdot dx_1 \tag{12}$$

Since x_1 is inspected, the portion of pdf outside of $[LL_1, UL_1]$ is truncated. The new truncated pdf needs to be multiplied by $1/p_1$ to normalize the integration of pdf back to 1. By replacing the original pdf with the truncated, the calculation of p_y becomes:

$$p_y = \int_{LL_1}^{UL_1} \frac{pdf(x_1)}{p_1}\cdot p_{y|x_1}(x_1)\cdot dx_1 \tag{13}$$

During the assembly process, any value of x_1 from the truncated pdf combining with $pdf(x_{2\text{-}4})$ can be treated as $pdf(x_{2\text{-}4})$ with a mean shift of x_1 as shown in Figure 6(a).

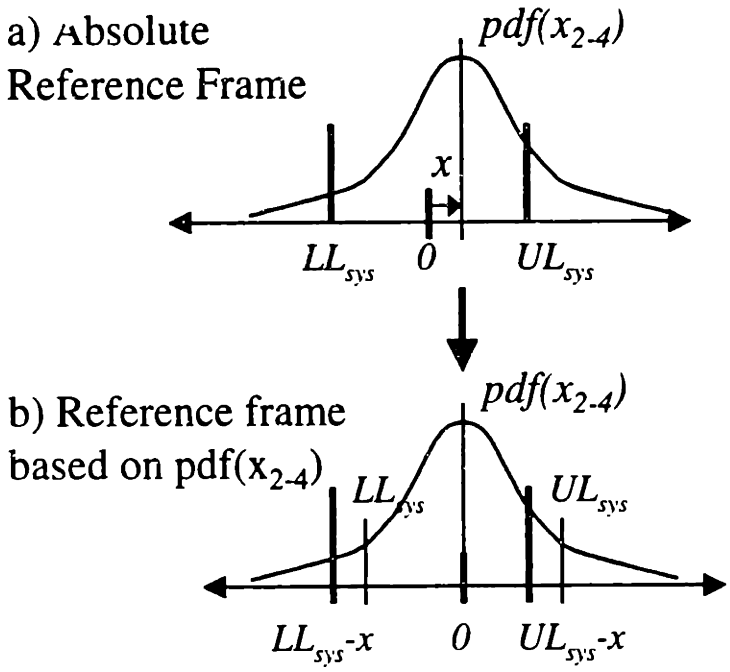


Figure 6: Shifting the Reference Frame to *pdf(x_{2-4})*

By shifting the reference frame to *pdf(x_{2-4})* as shown in Figure 6(b), the probability of producing an acceptable system for any given x_1 can be calculated by:

$$p_{y|x_1}(x_1) = \int_{LL_{n1}-x_1}^{UL_{n1}-x_1} pdf(x_{2-4}) \cdot dx_{2-4} \tag{14}$$

Combining Equation 13 and Equation 14, p_y can be calculated by:

$$p_y = \int_{LL_1}^{UL_1} \frac{pdf(x)}{p_1} \cdot \int_{LL_{n1}-x_1}^{UL_{n1}-x_1} pdf(x_{2-4}) \cdot dx_{2-4} \cdot dx \tag{15}$$

The plot of p_y as a function of inspection tolerance, t_1, is shown in Figure 7.

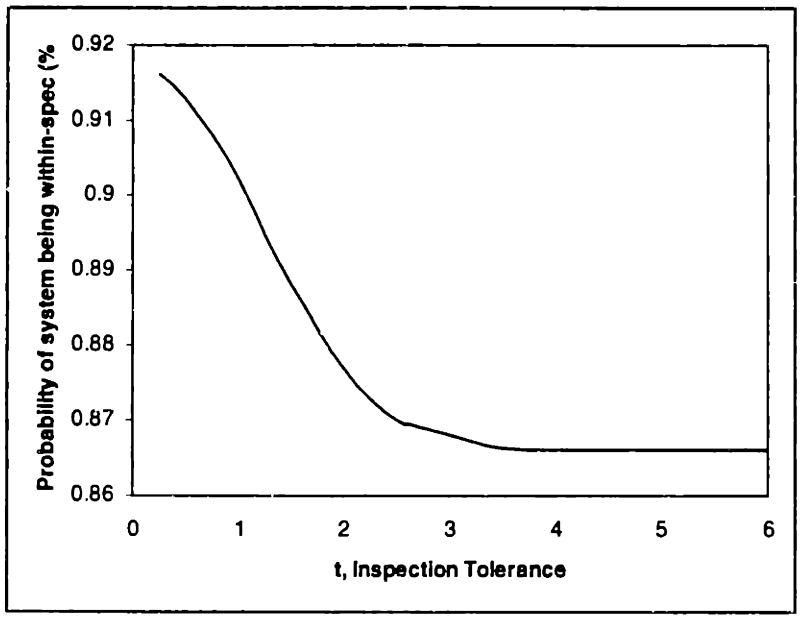


Figure 7: Plotting p_y as a Function of t_1, the Inspection Tolerance Setting for Part 1.

2.2.3 Summary and Result

Since the values for p_1 and p_y vary as the inspection limits, UL_1 and LL_1, change, like Equation 11, Equation 15 can be expressed as a function of t_1, which measures the inspection tolerance tightness. By calculating C_T with different values of t_1, we can optimize the inspection limits so that C_T is minimized.

$$C_T(t_1) = \frac{1}{p_1(t_1)} \cdot C_{i_1} + [1 - p_y(t_1)] \cdot C_f + \frac{1 - p_1(t_1)}{p_1(t_1)} \cdot C_{p_1} \qquad (16)$$

$C_T(t)$ can also be differentiated with respect to t_1 and the optimal inspection tightness for x_1 can be found by finding the t_1 value that minimizes the total expected cost. However, this equation is too complex to explicitly find the minimum. To facilitate finding the optimal solution, a pre-packaged mathematical software, *MathCad*, was used to perform the computation.

Table 1: System tolerance requirement, part process capability, and costs

System Requirement:	$[LL_{sys}, UL_{sys}] = \pm 3\sigma_1$
Process Capability:	$\sigma_1 = \sigma_2 = \sigma_3 = \sigma_4$

Costs:	C_m=20, C_i=0.1, C_p=1, C_f=20

Figure 8 shows the results of the cost calculation using the values in Table 1 and assuming no mean shift, and normalizing the inspection tolerances as a measure of t_I. The plot suggests inspecting the part at around 1.2 σ_I away from nominal is the optimal plan. Inspecting at tolerance limits tighter than 1.25 σ_I causes the total cost to increase significantly. The plot also shows the inspection tolerance making little difference when the inspection tolerance is set past approximately 3.4 standard deviation. This is consistent with statistics. When inspecting the population of a Gaussian distribution at ±3.4 standard deviation, only 0.06 percent of the population lies outside the inspection tolerance. In essence, inspecting at tolerance >3.4 standard deviation is the same as no inspection.

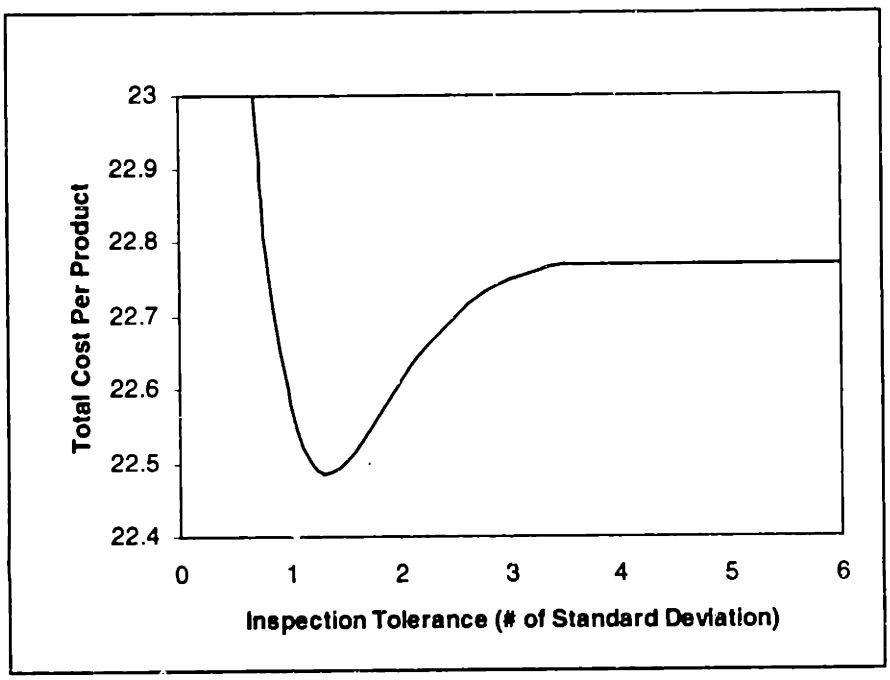


Figure 8: The Total Cost, C_T, Varies as t_I Varies.

2.3 Multiple Inspection

2.3.1 Inspect Two Parts

Often times, inspection is carried out at multiple sources of variation. We start the analysis for multiple inspections by extending the previous analysis from inspecting one part to inspecting two parts. When two out-of-four parts, part1 and part 2, are inspected, the probability tree becomes more complex (Figure

9). Instead of accepting and rejecting one part for single inspection, there are four outcomes for double inspection: accepting both parts, rejecting part 1, rejecting part 2 and rejecting both. The first two rejection scenarios can be represented as modules of single inspection with the "Inspect part 1" and the "Inspect part 2" modules represent selecting x_1 and new x_2 values until acceptable parts are found. Similar to the previous example, the last rejection scenario merely repeats the original decision branching process where part 1 and part 2 are inspected.

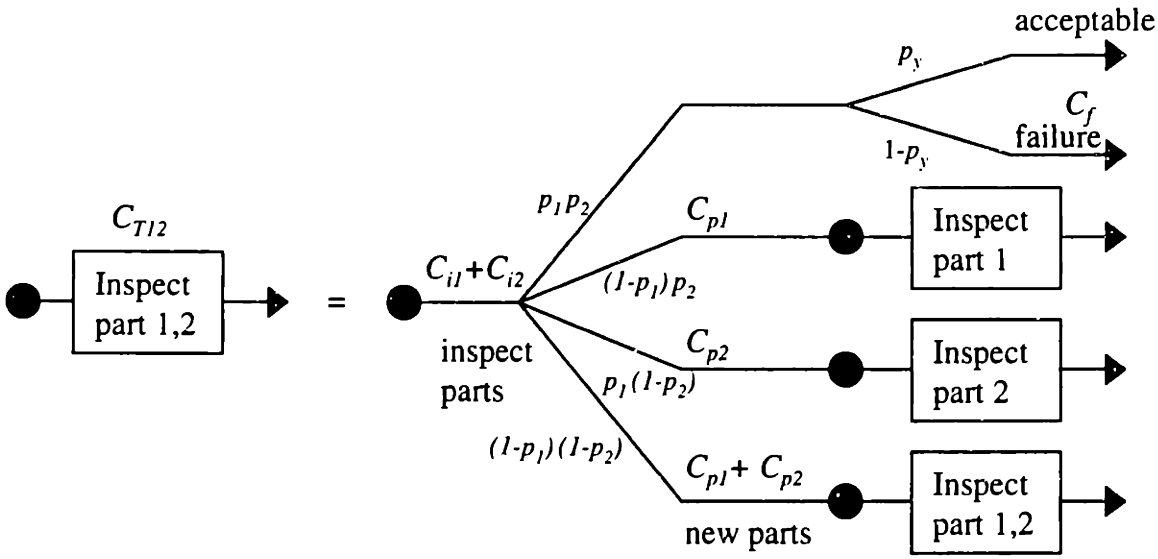


Figure 9: Decision Tree Modules for Inspecting Two Parts

Summing the cost of all the possible outcomes, the expected total cost of inspection becomes

$$C_{T12} = C_{i_1} + C_{i_2} + p_1 p_2 (1-p_y) C_f + (1-p_1) p_2 (C_{T_1} + C_{p_1}) + (1-p_2) p_1 (C_{T_2} + C_{p_2}) + (1-p_1)(1-p_2)(C_{T_{1,2}} + C_{p_1} + C_{p_2}) \quad (17)$$

C_{T1}, C_{T2}, and C_{T12} are the expected cost for modules of inspecting part 1, inspecting part 2, and inspecting both.

C_{T1} and C_{T2} are formulated using the result from the single-inspection analysis.

$$C_{T_1} = \frac{1}{p_1} \cdot C_{i_1} + (1-p_y) \cdot C_f + \frac{1-p_1}{p_1} \cdot C_{p_1} \quad (18)$$

$$C_{T_2} = \frac{1}{p_2} \cdot C_{i_2} + (1-p_y) \cdot C_f + \frac{1-p_2}{p_2} \cdot C_{p_2} \quad (19)$$

After simplifying algebraically, the final equation for inspecting two features becomes

$$C_T = \frac{1}{p_1} \cdot C_{i_1} + \frac{1}{p_2} \cdot C_{i_2} + (1 - p_y) \cdot C_f + \frac{1 - p_1}{p_1} \cdot C_{p_1} + \frac{1 - p_2}{p_2} \cdot C_{p_2} \tag{20}$$

where

$$p_y = \int_{LL_1}^{UL_1} \frac{pdf(x_1)}{p_1} \cdot \left[\int_{LL_2}^{UL_2} \frac{pdf(x_2)}{p_2} \cdot \left(\int_{LL_{3-4}-x_1-x_2}^{UL_{3-4}-x_1-x_2} pdf(x_{3-4}) \cdot dx_{3-4} \right) \cdot dx_2 \right] \cdot dx_1 \tag{21}$$

Plotting C_T as a function of inspection tolerance t_1 for part 1 and t_2 for part 2 yields the plot in Figure 10.

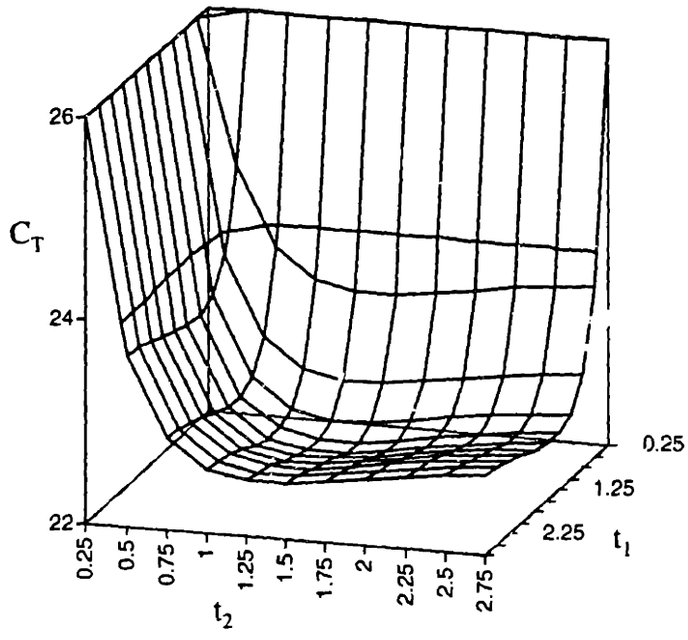


Figure 10: Contour Plot of C_T as a Function of Normalized Inspection Tolerances, t_1 and t_2.

When both part 1 and part 2 are inspected, there is an optimal inspection level for both where the total cost is minimal. The net cost is insensitive to small changes in the inspection limits. However, excessively tight or loose tolerances result in higher cost.

2.3.2 Inspect Three Parts

Following the previous analysis, the case of inspecting three out of the four features can be expanded as shown in Figure 11 below.

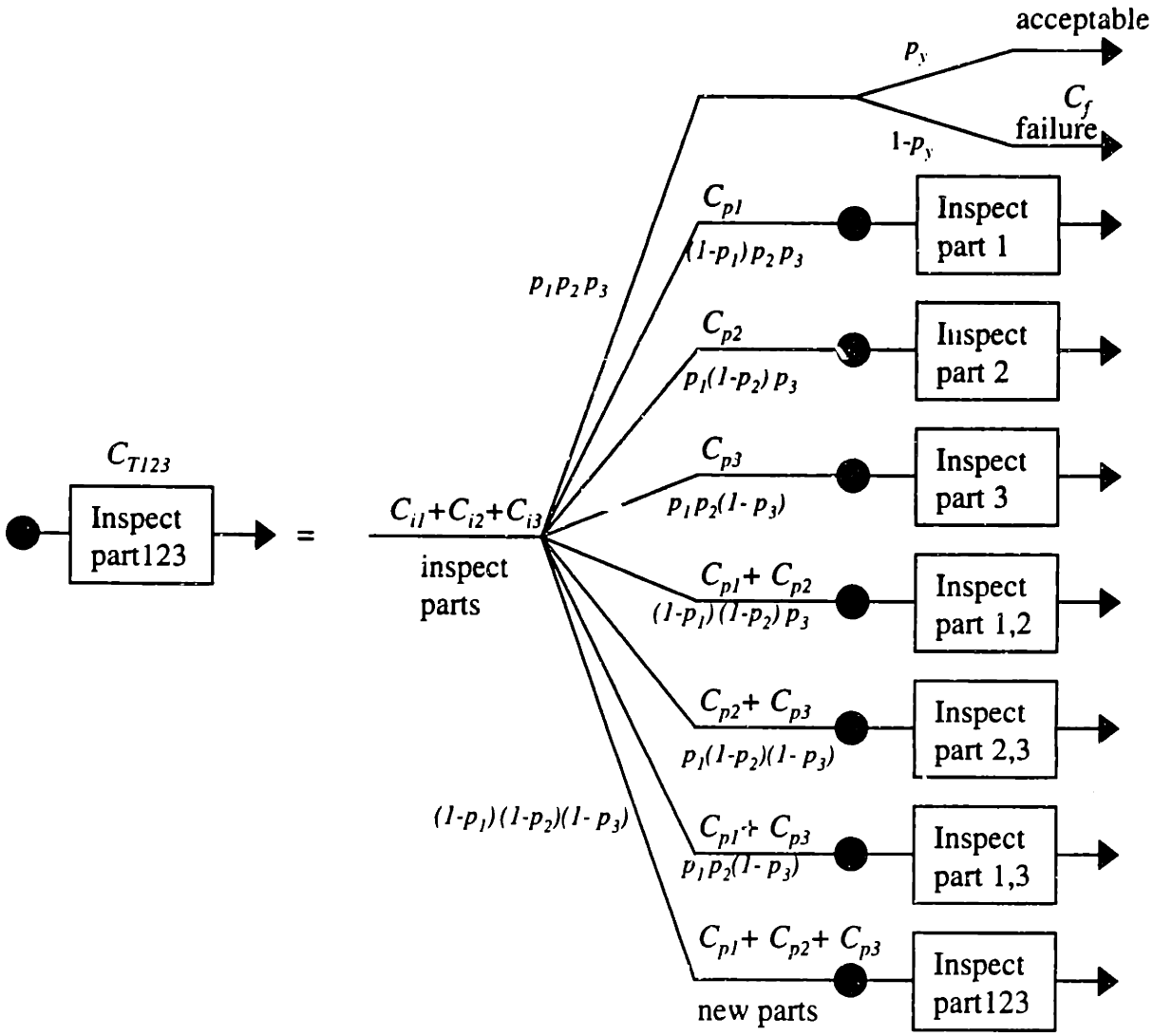


Figure 11: Decision Tree Modules for Inspecting Three Parts

In this case, there are eight possible combinations of accepting and rejecting the three inspected parts. The total expected cost, as calculated from the eight possible outcomes, is

$$\begin{aligned} C_{T_{123}} &= C_{i_1} + C_{i_2} + C_{i_3} + p_1p_2p_3(1-p_y)C_f \\ &+ (1-p_1)p_2p_3(C_{T_1} + C_{p_1}) + (1-p_2)p_1p_3(C_{T2} + C_{p_2}) + (1-p_3)p_1p_2(C_{T3} + C_{p_3}) \\ &+ p_3(1-p_1)(1-p_2)(C_{T_{12}} + C_{p_3} + C_{p_2}) + p_1(1-p_2)(1-p_3)(C_{T_{23}} + C_{p_2} + C_{p_3}) \\ &+ p_2(1-p_1)(1-p_3)(C_{T_{13}} + C_{p_1} + C_{p_3}) + (1-p_1)(1-p_2)(1-p_3)(C_{T_{123}} + C_{p_1} + C_{p_2} + C_{p_3}) \end{aligned} \tag{22}$$

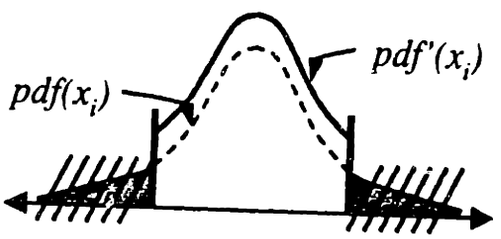
The equation can be simplified algebraically to

$$C_{T_{123}} = \frac{1}{p_1}\cdot C_{i_1} + \frac{1}{p_2}\cdot C_{i_2} + \frac{1}{p_3}\cdot C_{i_3} + (1-p_y)\cdot C_f + \frac{1-p_1}{p_1}\cdot C_{p_1} + \frac{1-p_2}{p_2}\cdot C_{p_2} + \frac{1-p_3}{p_3}\cdot C_{p_3} \tag{23}$$

where

$$p_y = \int_{LL_1}^{UL_1} \frac{pdf(x_1)}{p_1} \cdot \left\{ \int_{LL_2}^{UL_2} \frac{pdf(x_3)}{p_2} \left[\int_{LL_3}^{UL_3} \frac{pdf(x_3)}{p_3} \cdot \left(\int_{LL_{sys}-x_1-x_2-x_3}^{UL_{sys}-x_1-x_2-x_3} pdf(x_4) \cdot dx_4 \right) \cdot dx_3 \right] \cdot dx_2 \right\} \cdot dx_1 \qquad (24)$$

So far, we have only dealt with inspection scenarios where at least one feature is not inspected and we were able to use it as the basis for shifting the frame of reference and formulating our equations. If all four sources of variation are inspected, while the C_T calculation have additional C_i and C_p terms, the calculation of p_y becomes more complex, and $pdf(x_4)$ needs to be replaced with $pdf'(x_4)$, which is a truncated normal distribution normalized to 1 using $1/p_4$.as shown in Figure 12.



$$pdf'(x_i) = \begin{cases} \dfrac{pdf(x_i)}{p_i} & LL_i \le x_i \le UL_i \\ 0 & otherwise \end{cases}$$

Figure 12: The Truncated pdf

2.3.3 Generalized Equations

The analysis above can be formulated into generalized equation that can be applied to all cases. For the general equations, it is assumed that the assembly is composed of n parts and each part is a source of variation contributing to the final variation. The following sections first formulate the equations for when k ($k<n$) out of n features are inspected, then describe the equations for when all n sources of variations are inspected, and finally take the effect of non-unity variation propagation transfer coefficients into account.

Inspect k out of n Sources of Variation. From the above analysis, we can formulate a general equation to account for the cost of inspecting k out of n sources of variation.

$$C_T = \frac{1}{p_1} \cdot C_{i_1} + \frac{1}{p_2} \cdot C_{i_2} + \ldots + \frac{1}{p_k} \cdot C_{i_k} + (1 - p_y) \cdot C_f + \frac{1 - p_1}{p_1} \cdot C_{p_1} + \frac{1 - p_2}{p_2} \cdot C_{p_2} + \ldots + \frac{1 - p_k}{p_k} \cdot C_{p_k} \qquad (25)$$

where

$$p_i = \int_{LL_i}^{UL_i} pdf(x_i)\cdot dx_i \tag{26}$$

and

$$p_y = \int_{LL_1}^{UL_1} \frac{pdf(x_1)}{p_1}\left\langle \int_{LL_2}^{UL_2} \frac{pdf(x_2)}{p_2}\cdot\left\{\cdots\left[\int_{LL_k}^{UL_k} \frac{pdf(x_k)}{p_k}\cdot\left(\int_{LL_{n-1}-x_1-x_2-\ldots-x_k}^{UL_{n-1}-x_1-x_2-\ldots-x_k} pdf(z)\cdot dz\right)\cdot dx_k\right]\cdots\right\}\cdot dx_2\right\rangle\cdot dx_1 \tag{27}$$

where all uninspected features are combined to form one new normal distribution, *pdf(z)*.

Inspect All *n* Sources of Variation. If inspection is carried out at all *n* sources of variation, the total cost of the inspection plan becomes

$$C_T = \frac{1}{p_1}\cdot C_{i_1} + \frac{1}{p_2}\cdot C_{i_2} + \ldots + \frac{1}{p_n}\cdot C_{i_n} + (1-p_y)\cdot C_f + \frac{1-p_1}{p_1}\cdot C_{p_1} + \frac{1-p_2}{p_2}\cdot C_{p_2} + \ldots + \frac{1-p_n}{p_n}\cdot C_{p_n} \tag{28}$$

since $k=n$.

Likewise, p_y can be calculated using

$$p_y = \int_{LL_1}^{UL_1} \frac{pdf(x_1)}{p_1}\cdot\left\langle \int_{LL_2}^{UL_2} \frac{pdf(x_2)}{p_2}\cdot\left\{\cdots\left[\int_{LL_{n-1}}^{UL_{n-1}} \frac{pdf(x_{n-1})}{p_{n-1}}\cdot\left(\int_{LL_{n-1}-x_1-x_2-\ldots-x_{n-1}}^{UL_{n-1}-x_1-x_2-\ldots-x_{n-1}} pdf'(x_n)\cdot dx_n\right)\cdot dx_{n-1}\right]\cdots\right\}\cdot dx_2\right\rangle\cdot dx_1 \tag{29}$$

where $pdf'(x_n)$ is the pdf of the truncated distribution, which is the unscrapped portion of the original distribution multiplied by $1/p_n$.

Non-unity variation propagation. The above analysis is limited to geometric stack-up problems. For non-stack-up geometric issues such as angle and non-geometric issues such as electromagnetic, electrical, and thermodynamic problems, Equation 29 is not general enough to be used in all cases. Therefore, an analysis that utilizes non-unity variation propagation is needed. For this purpose, the following equation provides the basis for analysis:

$$\Delta y = c_1\cdot\Delta x_{12} + c_2\cdot\Delta x_{22} + \ldots + c_n\cdot\Delta x_n \tag{30}$$

Since the transfer coefficients, c_i, do not affect inspections at the part level, the equation for p_i and the cost equation stays the same. Only p_y needs to be modified to accommodate the coefficients.

For the case where only k out of n features are inspected, Equation 29 becomes:

$$p_y = \int_{LL_1}^{UL_1} \frac{pdf(x_1)}{p_1} \cdot \left\langle \int_{LL_2}^{UL_2} \frac{pdf(x_2)}{p_2} \cdot \left\{ \dots \cdot \left[\int_{LL_k}^{UL_k} \frac{pdf(x_k)}{p_k} \cdot \left(\int_{LL_{n+1}-c_1x_1-c_2x_2-\dots-c_kx_k}^{UL_{n+1}-c_1x_1-c_2x_2-\dots-c_kx_k} pdf(z) \cdot dz \right) \cdot dx_k \right] \cdot \dots \right\} \cdot dx_2 \right\rangle \cdot dx_1 \quad (31)$$

where *pdf(z)* is the probability density function created by lumping together all uninspected pdfs.

If those uninspected pdfs are normal distributions *pdf(z)* can be calculated using:

$$pdf(z) = \frac{1}{\sqrt{2\pi}\sigma_z} e^{\frac{(z-m)^2}{2\sigma_z^2}} \quad (32)$$

and

$$\sigma_z = \sqrt{c_{k+1}\sigma_{k+1}^2 + c_{k+2}\sigma_{k+2}^2 + \dots + c_n\sigma_n^2} \quad (33)$$

However, if one or many uninspected pdfs are not normal distributions, *pdf(z)* needs to be calculated explicitly and its formulation is discribed in next Section.

In the case where all n features are inspected, p_y can be calculated using:

$$p_y = \int_{LL_1}^{UL_1} \frac{pdf(x_1)}{p_1} \cdot \left\langle \int_{LL_2}^{UL_2} \frac{pdf(x_2)}{p_2} \cdot \left\{ \dots \cdot \left[\int_{LL_{n-1}}^{UL_{n-1}} \frac{pdf(x_{n-1})}{p_{n-1}} \cdot \left(\int_{\frac{LL_{n+1}-c_1x_1-c_2x_2-\dots-c_{n-1}x_{n-1}}{c_n}}^{\frac{UL_{n+1}-c_1x_1-c_2x_2-\dots-c_{n-1}x_{n-1}}{c_n}} pdf'(x_n) \cdot dx_n \right) \cdot dx_{n-1} \right] \cdot \dots \right\} \cdot dx_2 \right\rangle \cdot dx_1 \quad (34)$$

where *pdf'(x_n)* is the pdf of truncated distribution, which is a truncated distribution multiplied by $1/p_n$.

2.4 Probability Density Function Propagation

While the above sections describe a methodology for analyzing multiple-inspection problems, it does not address the issue of variation propagating through multiple layers. To deal with this issue, we need an equation taking the inspected and uninspected pdfs from the lower level and propagate the distribution to form a new pdf at the next level. This pdf propation method can be applied several times so all pdfs at the lowest layer can be propagated to the next layer and ultimately the generalized equations in Section 2.3.3 can be used to calculate the total cost at the top level.

2.4.1 Two-Part Assembly

To start our formulation, we first look at examples of simple probability mass functions (pmfs). If $y=x_1+x_2$, and x_1 and x_2 have pmf shown in Figure 13,

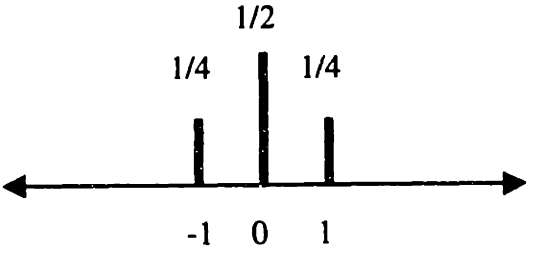


Figure 13: Pmf of x_1 and x_2

we can calculate the pmf for y using Equation 35 as follows and plot it in Figure 14:

$$\begin{aligned}
pmf_y(2) &= pmf_{x_1}(1)\cdot pmf_{x_2}(1)+ pmf_{x_1}(0)\cdot pmf_{x_2}(2)+ pmf_{x_1}(-1)\cdot pmf_{x_2}(3)= 1/16 \\
pmf_y(1) &= pmf_{x_1}(1)\cdot pmf_{x_2}(0)+ pmf_{x_1}(0)\cdot pmf_{x_2}(1)+ pmf_{x_1}(-1)\cdot pmf_{x_2}(2)= 1/4 \\
pmf_y(0) &= pmf_{x_1}(1)\cdot pmf_{x_2}(-1)+ pmf_{x_1}(0)\cdot pmf_{x_2}(0)+ pmf_{x_1}(-1)\cdot pmf_{x_2}(1)= 3/8 \\
pmf_y(-1) &= pmf_{x_1}(1)\cdot pmf_{x_2}(-2)+ pmf_{x_1}(0)\cdot pmf_{x_2}(-1)+ pmf_{x_1}(-1)\cdot pmf_{x_2}(0)= 1/4 \\
pmf_y(-2) &= pmf_{x_1}(1)\cdot pmf_{x_2}(-3)+ pmf_{x_1}(0)\cdot pmf_{x_2}(-2)+ pmf_{x_1}(-1)\cdot pmf_{x_2}(-1)= 1/16
\end{aligned} \tag{35}$$

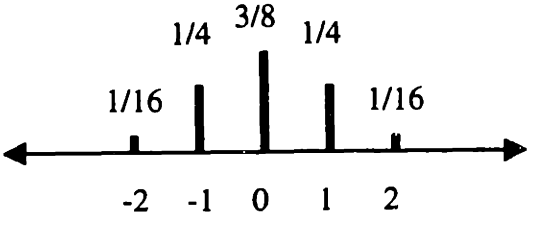


Figure 14: Pmf of y

The calculation can be formulated as:

$$pmf_y(y_0) = \sum_{x_1} \left[pmf_{x_1}(x_1)\cdot pmf_{x_2}(y_0 - x_1) \right] \tag{36}$$

When x_1 and x_2 are continuous Gaussian pdfs, we can extend the pmf equation to a pdf equation:

$$pdf_y(y_0) = \int_{-\infty}^{\infty} [pdf_{x_1}(x) \cdot pdf_{x_2}(y_0 - x)] \cdot dx \qquad (37)$$

To accommodate the input of x_1 and x_2 having a truncated pdf due to the result of inspection and scrapping, we uses a new notation for the equation

$$pdf_y(y_0) = \int_{-\infty}^{\infty} [pdf_{x_1}'(x) \cdot pdf_{x_2}'(y_0 - x)] \cdot dx \qquad (38)$$

where

If x_i is not inspected	$pdf_i'(x_0) = pdf_i(x_0) \qquad -\infty < x_0 < \infty$ (39) => original pdf
If x_i is inspected	$pdf_i'(x_0) = \begin{cases} \dfrac{pdf_i(x_0)}{p_i} & LL_i \le x_0 \le UL_i \\ 0 & otherwise \end{cases}$ (40) => truncated pdf

Therefore, if x_1 is inspected, we can change the equation to

$$pdf_y(y_0) = \int_{LL_1}^{UL_1} \left[\frac{pdf_{x_1}(x)}{p_1} \cdot pdf_{x_2}'(y_0 - x) \right] \cdot dx \qquad (41)$$

When x_1 is inspected and x_2 is not, the pdf of y is shown in Figure 15. The plot shows the pdf comparing to the case of no inspection and the case of x_1 having perfect quality (all variation of x_1). The pdf of y changes from the original uninspected pdf to the best possible pdf as the inspection limits for x_1 [LL_1, UL_1] move from $\pm\infty$ to 0, or from no inspection to perfect quality. If the system tolerance specification is [-2.0,+2.0], then the probability of failure is 15.7% for no inspection, 7.8% for inspecting x_1 at ± 1.0, and 4.6% for inspecting x_1 at perfect quality.

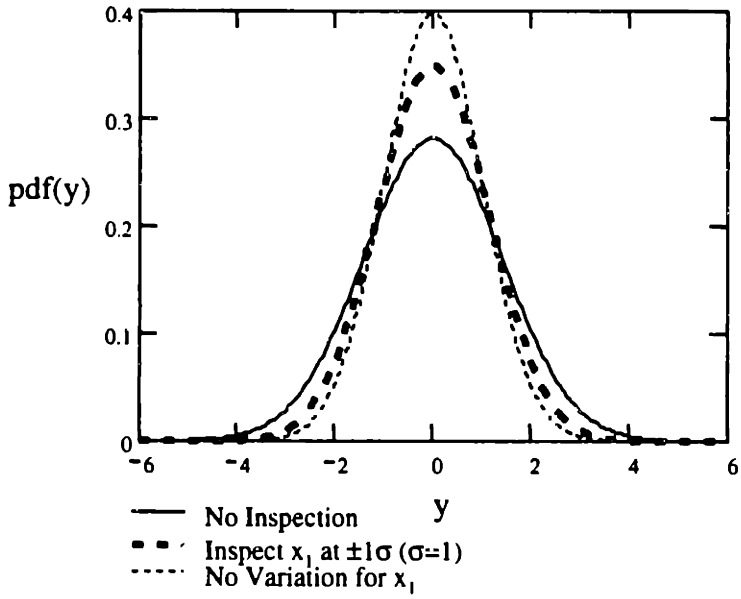


Figure 15: Pdf of y when x_1 is Inspected at ±1

When both x_1 and x_2 are inspected, both pdfs are truncated. The resultant *pdf(y)* is shown in Figure 16. If the system specification is [-2.0, +2.0], comparing to the failure rate of 15.7% for no inspection and 7.8% for inspecting x_1, inspect both x_1 and x_2 results in a failure rate of 0%. Inspecting both x_1 and x_2 at [-1.0, +1.0] is shown to yield a lower failure rate than eliminating all variation of x_1. Therefore, it is a more effective inspection plan.

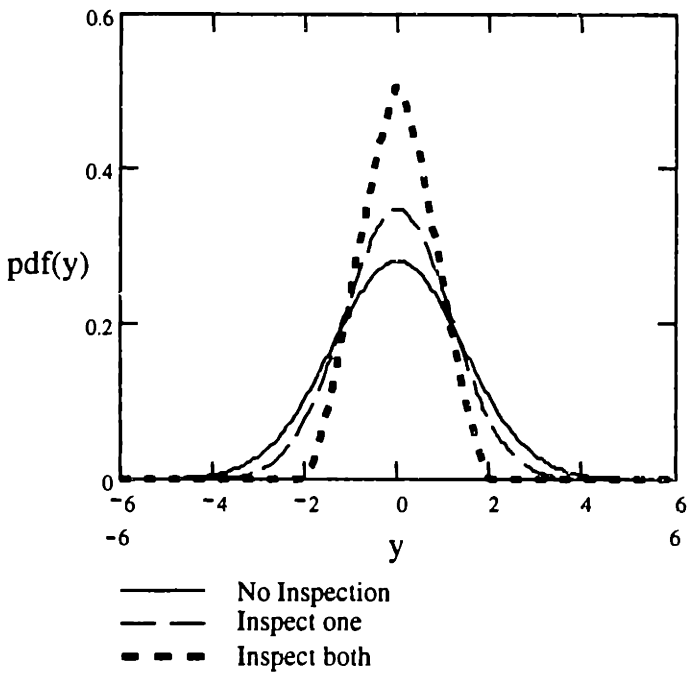


Figure 16: Pdf of y when both x_1 and x_2 are inspected at ±1

2.4.2 Three-Part Assembly

Likewise, if we extend the scenario to $y=x_1+x_2+x_3$, the equation becomes

$$pmf_y(y_0) = \sum \left\{ pmf_1(x_1) \cdot \sum \left[pmf_2(x_2) \cdot pmf_3(y_0 - x_1 - x_2) \right] \right\} \tag{42}$$

If x_1, x_2, and x_3 have continuous pdfs, we can express the pdf of y as

$$pdf_y(y_0) = \int_{-\infty}^{\infty} \left\{ pdf_1(x_1) \cdot \int_{-\infty}^{\infty} \left[pdf_2(x_2) \cdot pdf_3(y_0 - x_1 - x_2) \right] \cdot dx_2 \right\} \cdot dx_1 \tag{43}$$

2.4.3 General Multi-Part Assembly

As a result, we can follow the same process and extend the equation for y to a general case where $y=x_1+x_2+\ldots+x_n$.

$$pdf_y(y_0) = \int_{-\infty}^{\infty} \left\langle pdf_1(x_1) \cdot \int_{-\infty}^{\infty} \left\{ pdf_2(x_2) \cdots \int_{-\infty}^{\infty} \left[pdf_{n-1}(x_{n-1}) \cdot pdf_n(y_0 - x_1 - x_2 - \ldots - x_{n-1}) \right] \cdot dx_{n-1} \right\} \cdots \cdot dx_2 \right\rangle \cdot dx_1 \tag{44}$$

Ultimately, if the scenario is the most general variation propagation problems with non-unity variation propagation coefficients ($y=c_1x_1+c_2x_2+\ldots+c_nx_n$ where y and x_i are variations measured relative to nominal), we get

$$pdf_y(y_0) = \int_{-\infty}^{\infty} \left\langle pdf_1(x_1) \cdot \int_{-\infty}^{\infty} \left\{ pdf_2(x_2) \cdots \int_{-\infty}^{\infty} \left[pdf_{n-1}(x_{n-1}) \cdot pdf_n\left(\frac{y_0 - c_1x_1 - c_2x_2 - \ldots - c_{n-1}x_{n-1}}{c_n} \right) \right] \cdot dx_{n-1} \right\} \cdots \cdot dx_2 \right\rangle \cdot dx_1 \tag{45}$$

Performing the multiple-integration is possible; however, it is not an easy task and takes heavy computational time.

2.5 Summary

So far, we have formulated equations for calculating the total cost of the inspection plan, for modeling the effect of multiple inspections on multiple sources of variations and, for propagating pdfs from the bottom layer to the top layer.

1. Total cost equation:

$$C_T = \frac{1}{p_1}\cdot C_{i_1} + \frac{1}{p_2}\cdot C_{i_2} + \ldots + \frac{1}{p_n}\cdot C_{i_n} + (1-p_1)\cdot C_f + \frac{1-p_1}{p_1}\cdot C_{p_1} + \frac{1-p_2}{p_2}\cdot C_{p_2} + \ldots + \frac{1-p_n}{p_n}\cdot C_{p_n} \qquad (46)$$

2. Effect of inspections on final quality equation:

$$p_1 = \int_{LL_1}^{UL_1} \frac{pdf(x_1)}{p_1}\cdot\left\langle \int_{LL_2}^{UL_2} \frac{pdf(x_2)}{p_2}\cdot\left\{\ldots\cdot\left[\int_{LL_{n-1}}^{UL_{n-1}} \frac{pdf(x_{n-1})}{p_{n-1}}\cdot\left(\int_{\frac{LL_{n1}-c_1x_1-c_2x_2-\ldots-c_{n-1}x_{n-1}}{c_n}}^{\frac{UL_{n1}-c_1x_1-c_2x_2-\ldots-c_{n-1}x_{n-1}}{c_n}} pdf(x_n)\cdot dx_n\right)\cdot dx_{n-1}\right]\cdot\ldots\right\}\cdot dx_2\right\rangle\cdot dx_1 \qquad (47)$$

Variation Propagation equation:

$$pdf_y(y_0) = \int_{-\infty}^{\infty}\left\langle pdf_1(x_1)\cdot\int_{-\infty}^{\infty}\left\{pdf_2(x_2)\cdot\ldots\int_{-\infty}^{\infty}\left[pdf_{n-1}(x_{n-1})\cdot pdf_n\left(\frac{y_0-c_1x_1-c_2x_2-\ldots-c_{n-1}x_{n-1}}{c_n}\right)\right]\cdot dx_{n-1}\right\}\cdot\ldots\cdot dx_2\right\rangle\cdot dx_1 \qquad (48)$$

The three equations can be utilized to solve the problem of finding optimal inspection settings for any multi-layer variation propagation scenarios.

CHAPTER 3: INSPECTION MODELING

3.1 Introduction

While the analysis in Chapter 2 can be used to find the optimum inspection tolerance for a multi-layer assembly, the problem can become more complex. Several issues make it difficult to apply the above analysis to a complex product.

1. Computing the multiple integrals to solve the equations explicitly takes enormous amount of computational capability

2. Complex products have several assembly layers, multiple product-KCs, and many part-KCs. Although the analysis in Section 2.4 showed it is possible to convert the multiple layers into a single layer by expressing the variations at the top level as functions of all contributing input variables, the conversion and calculation becomes cumbersome for complex products.

3. While products, components, and processes have costs associated with them, the multiple KCs in a single product or single compo: ,nt cannot be assigned separate cost and it is difficult to divide costs among the multiple product-KCs.

4. A single part or subassembly may contain multiple part-KCs each contributing to multiple product-KCs. Scrapping a part based on its contribution to product-KC i changes the distribution for product-KC j as well.

5. Scrapping a part and reworking a feature changes the variation distribution in different ways.

6. After a feature fails an inspection test, rework will reduce variation for individual features, but scrapping impacts all features contained in the scrapped part. While rework can be focused on individual features, scrapping can't.

Because of the complexity, a general model which captures both the product assembly sequence and the variation propagation and implements a simulation-based analysis for inspection and corrective action was developed. For the above reasons, the *KCTool* software modeling backbone is used as the basis for modeling both the product assembly sequence and the KC flowdown of different products. The general *KCTool* model is shown in Figure 17.

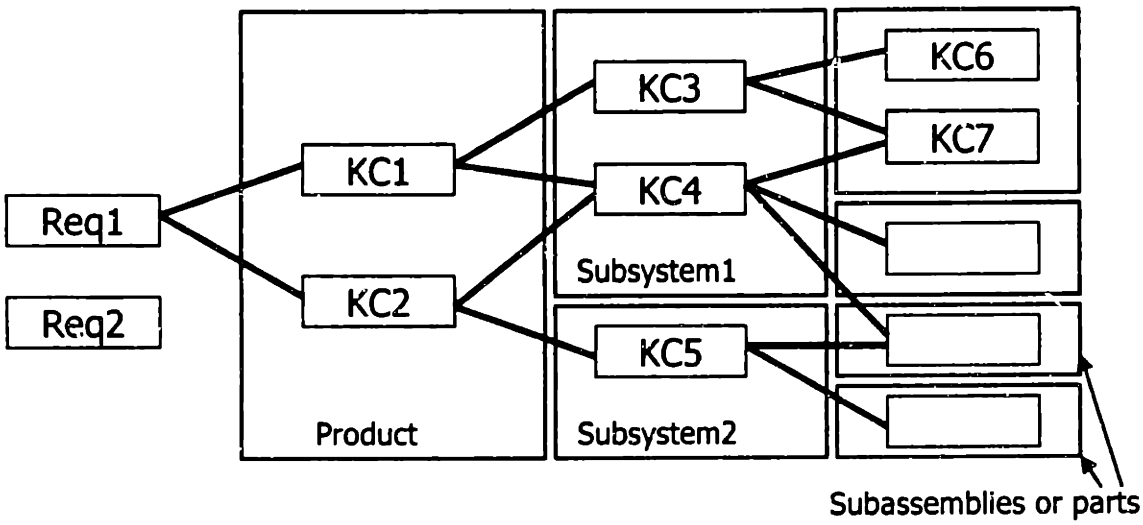


Figure 17: Modeling Product Decomposition and KC Flowdown in *KCTool*

The *KCTool* model contains two superimposed models: a product assembly model and a KC flowdown model. Section 3.2 describes the product model and Section 3.3 describes the KC variation propagation model, Section 3.4 describes the process for deriving the transfer coefficients, and Section 3.5 describes the software modeling features.

3.2 Assembly Plan Model

For a complex product, individual components are manufactured and then assembled into subassemblies; subassemblies are then assembled into subsystems and then the final product. The product assembly sequence varies depending on the product.

The product assembly model captures the order in which physical components, subassemblies, and subsystems are assembled. Figure 18 shows the assembly process for a four-part product. First, Part 1 and 2 are assembled into Subsystem 1 and Part 3 and 4 are assembled into Subsystem 2. Then, Subsystem 1 and Subsystem 2 are assembled into the final product. The software allows multiple assembly layers to be captured with no limitation on the number of layers and the number of parts. In addition, variation induced by manufacturing or assembly process that introduces variation can also be modeled as another component going into an assembly. This modeling scheme can also model secondary operations performed after assembly by adding another layer and treating the secondary operations as components going into the added assembly layer.

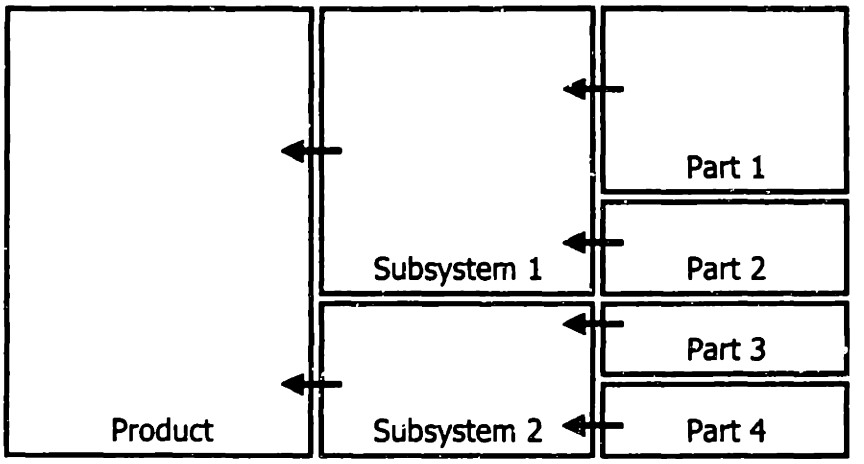


Figure 18: Product Assembly Plan

The individual parts and assembly processes are each assigned a unit cost. The software automatically calculates the total costs in each assembly step by summing the component costs and adding an assembly cost. During simulation, when a part is scrapped, its cost is added to the total cost. When a sub-assembly is scrapped, the par costs and the assembly cost are added to the total cost.

3.3 KC Variation Propagation Model

The KC relationships are maintained in a tree shown in Figure 19 where the individual nodes may point up to more than one parent KC and each parent KC may have more than one child. This many-to-many relationship is essential to capture the complexity of most systems.

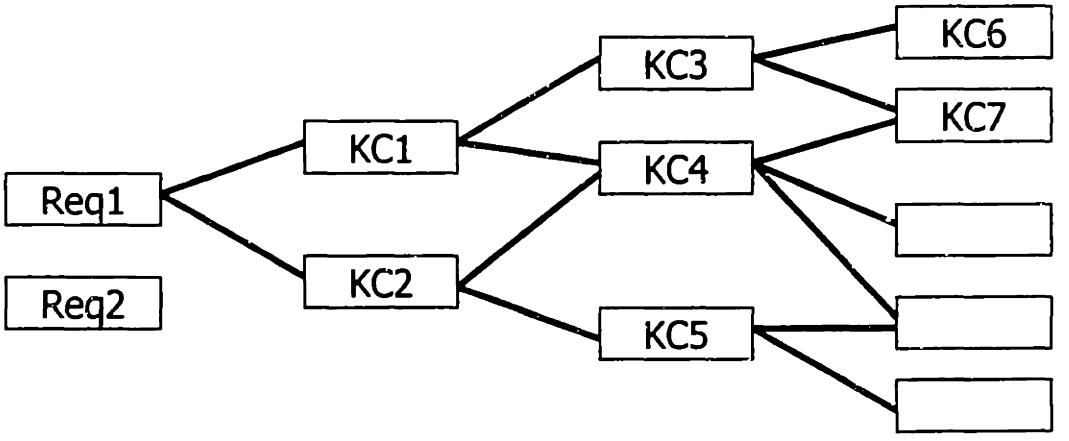


Figure 19: Tracing though the KC Variation Propagation Model

Formulation of the variation propagation was done by Thornton [1998]. The following model calculates the product-KC's standard deviation from the process-KC's standard deviation based on a linearized

variation model. More detailed and computationally intense models such as Variation Systems Analysis (VSA) can be used to predict a more accurate capability.

Figure 20 is a generalized version of the KC flowdown in Figure 19. The top layer of the flowdown has n_f product-KCs. Each has a nominal value f_i, a tolerance $[LL_i, UL_i]$, a standard deviation, σ_i, a mean, μ_i, and a failure cost, C_i. Product-KCs are created by ℓ layers of subsystem-, process-KCs, x_{jk} (the j^{th} parameter in the k^{th} level). Each layer, k, has n_k KCs.

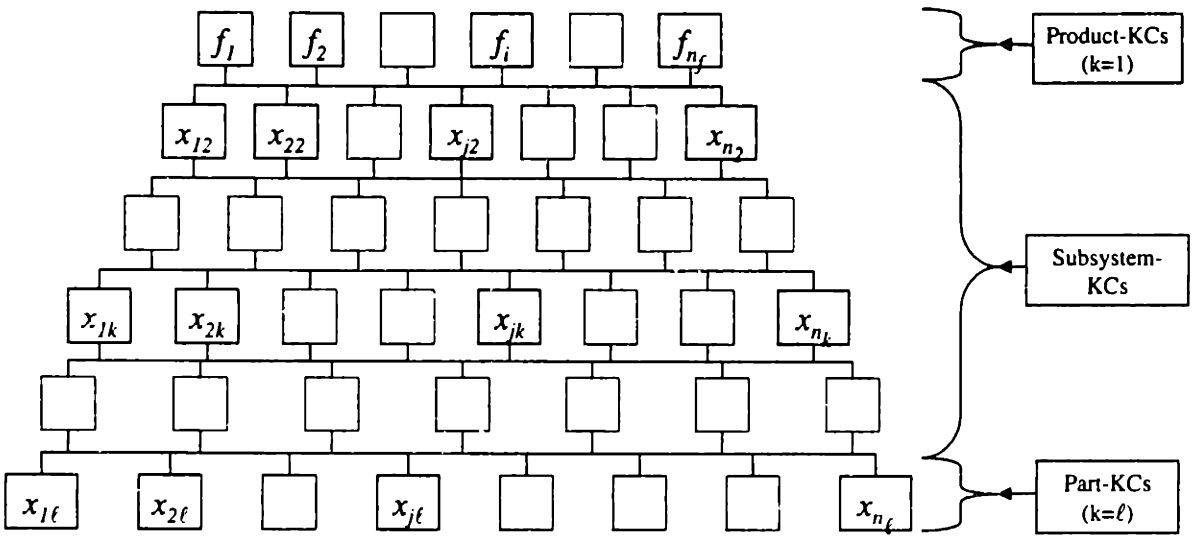


Figure 20: Generalized KC Flowdown

By Taylor expansions, the expression relating deviations in a product-KC, Δf_i[4] to the next KC layer, $\Delta \mathbf{x_2}$ is represented as a linear equation.

$$\Delta f_i = \frac{\partial f_i}{\partial x_{12}}\Delta x_{12} + \frac{\partial f_1}{\partial x_{22}}\Delta x_{22} + \ldots + \frac{\partial f_1}{\partial x_{n_2 2}}\Delta x_{n_2 2} \qquad (49)$$

$\partial f_i/\partial x_{j2}$ captures the sensitivity of a product-KC, f_i, to a change in subsystem-KC, Δx_{j2} (termed variation-sensitivity). The sensitivities can be derived using DOE [Phadke 1989], tolerance chains [Cunningham, Mantripragada et al. 1996], or modeling [Frey, Otto et al. 1998]. The array subsystem-KC deviations, $\Delta \mathbf{x_2}$, can be related to the product-KCs array, $\Delta \mathbf{f}$, through the matrix of partial derivatives, δ_1. A matrix

[4] $\Delta f = \Delta x_1$, but we are using f to be consistent with other literature [Srinivasan 1998].

approach to variation propagation and modeling has been used extensively by authors such as Homann and Thornton [1998], Slocum [1992], Suh [1990], Frey, *et al.* [1997] and Gao *et al.* [1998].

$$\mathbf{\Delta f} = \mathbf{\delta_1 \Delta x_2} \text{ where } \mathbf{\delta_1} = \begin{bmatrix} \frac{\partial f_1}{\partial x_{12}} & \frac{\partial f_1}{\partial x_{22}} & \cdots & \frac{\partial f_1}{\partial x_{n_2 2}} \\ \frac{\partial f_2}{\partial x_{12}} & \frac{\partial f_2}{\partial x_{22}} & \cdots & \frac{\partial f_2}{\partial x_{n_2 2}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_{n_1}}{\partial x_{12}} & \frac{\partial f_{n_1}}{\partial x_{22}} & \cdots & \frac{\partial f_{n_1}}{\partial x_{n_2 2}} \end{bmatrix} \tag{50}$$

For each subsequent layer, δ_1 can be generalized to the matrix, δ_k, which relates $\Delta \mathbf{x_{k+1}}$ to $\Delta \mathbf{x_k}$.

$$\mathbf{\Delta x_K} = \mathbf{\delta_K \Delta x_{(K+1)}}, \text{ where } \mathbf{\delta_K} = \begin{bmatrix} \frac{\partial x_{1k}}{\partial x_{1(k+1)}} & \frac{\partial x_{1k}}{\partial x_{2(k+1)}} & \cdots & \frac{\partial x_{1k}}{\partial x_{n_{k+1}(k+1)}} \\ \frac{\partial x_{2k}}{\partial x_{1(k+1)}} & \frac{\partial x_{2k}}{\partial x_{2(k+1)}} & \cdots & \frac{\partial x_{2k}}{\partial x_{n_{k+1}(k+1)}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial x_{n_k k}}{\partial x_{1(k+1)}} & \frac{\partial x_{n_k k}}{\partial x_{2(k+1)}} & \cdots & \frac{\partial x_{n_k k}}{\partial x_{n_{k+1}(k+1)}} \end{bmatrix} \tag{51}$$

The correlation between variation in process-KCs (represented by array $\mathbf{x_1}$) and product-KCs can be expressed by:

$$\mathbf{\Delta f} = \mathbf{\delta_1 \delta_2 \delta_3 \ldots \delta_{\ell-1} \Delta x_\ell} \textit{ where } \mathbf{D} = \mathbf{\delta_1 \delta_2 \delta_3 \ldots \delta_{(\ell-1)}} \tag{52}$$

The analysis in Equations 49-52 models errors for a single product. In addition, the statistical behavior of errors can be modeled using a similar analysis. The partial differentials are used to calculate standard deviation, σ_{jk} for any KC, x_{jk}.

$$\sigma_{jk}^2 = \left(\frac{\partial x_{jk}}{\partial x_{1(k+1)}}\right)^2 \sigma_{1(k+1)}^2 + \left(\frac{\partial x_{jk}}{\partial x_{2(k+1)}}\right)^2 \sigma_{2(k+1)}^2 + \ldots + \left(\frac{\partial x_{jk}}{\partial x_{n_{(k+1)}(k+1)}}\right)^2 \sigma_{n_{(k+1)}(k+1)}^2 \tag{53}$$

Equations 53 can also be represented in a matrix form but a second matrix, τ_k, is needed to calculate standard deviations.

$$\tau_K = \begin{bmatrix} \left(\frac{\partial x_{1k}}{\partial x_{1(k+1)}}\right)^2 & \left(\frac{\partial x_{1k}}{\partial x_{2(k+1)}}\right)^2 & \cdots & \left(\frac{\partial x_{1k}}{\partial x_{n_{k+1}(k+1)}}\right)^2 \\ \left(\frac{\partial x_{2k}}{\partial x_{1(k+1)}}\right)^2 & \left(\frac{\partial x_{2k}}{\partial x_{2(k+1)}}\right)^2 & \cdots & \left(\frac{\partial x_{2k}}{\partial x_{n_{k+1}(k+1)}}\right)^2 \\ \vdots & \vdots & \vdots & \vdots \\ \left(\frac{\partial x_{n_k k}}{\partial x_{1(k+1)}}\right)^2 & \left(\frac{\partial x_{n_k k}}{\partial x_{2(k+1)}}\right)^2 & \cdots & \left(\frac{\partial x_{n_k k}}{\partial x_{n_{k+1}(k+1)}}\right)^2 \end{bmatrix} \quad (54)$$

When Equations 53 and 54 are combined, the product variation can be predicted.

$$\sigma_f^2 = \mathbf{T}\sigma_\ell^2 \; where \mathbf{T} = \tau_1\tau_2\tau_3 \ldots \tau_{(\ell-1)} \quad (55)$$

where σ_f is the array of product-KC standard deviation and σ_ℓ is the array of the process- and part- KCs standard deviations. In most cases $D^2_{ij}=T_{ij}$ except when a part- or process-KC contributes multiple times to the same product-KC.[5]

3.4 Derivation of Transfer Coefficients

The application of the variation propagation model is shown in the following industry case study (exact component geometry are masked to protect proprietary information). High-speed air jet blows from nozzle at point A to the angled plate at point B. The distance, ΔL, from the leading edge to air stream is a key characteristics and is essential for the system performance.

[5] This may happen, for example, if a single worn tool is used to make multiple parts for an assembly and, consequently, all parts in the assembly are undersized.

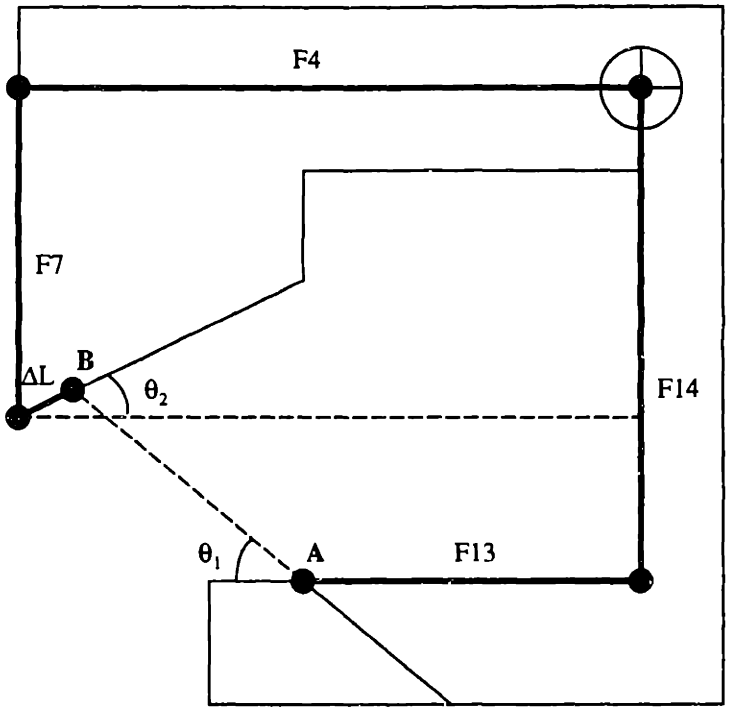


Figure 21: Geometry of Airjet Stream

Using geometry, the value of ΔL can be determined using

$$\Delta L = \frac{F14 - F7 - (F4 - F13)\cdot \tan\theta_1}{\sin\theta_2 - \cos\theta_2 - \tan\theta_1} \tag{56}$$

By differentiating with respect to the individual variables and substituting the nominal values of θ_1=28.5 degrees and θ_2=3.60 degrees, the above equation can be converted to a transfer function of variation propagation which the same format as Equation 49.

$$\Delta L = 0.043\cdot\Delta 6_1 - 0.0085\cdot\Delta 6_2 - 0.8979\cdot\Delta F4 + 1.6539\cdot\Delta F7 - 0.8979\cdot\Delta F13 + 1.6538\cdot\Delta F14 \tag{57}$$

3.5 *KCTool* Software Features

Using the *KCTool* software, the KC variation propagation hierarchy is superimposed on the product decomposition to capture the relationship between the physical components and the multiple KCs on the parts.

A user executes the *KCTool* software by running the *kctool.exe* file. The information for a *KCTool* model is saved in an Access database file. The graphical user interface (GUI) allows engineers to model both the product decomposition and the KC variation propagation hierarchy. In the *KCTool* window environment,

dividing the boxes horizontally creates parallel parts and dividing them vertically creates assembly stages. Drawing a KC node inside any component box creates a KC. The software keeps track of which KCs belong to which component or subsystem. The user is free to move KCs to different parts, and the software updates the changes. By drawing a connection line between two KCs, the user can create many-to-many variation propagation contributions. These contributions are essential during the simulation process to model the variation propagation.

Clicking on the parts, the KCs, and the lines connecting the KCs can access additional information. When the user clicks on a part, a new window pops up. The window contains information for part costs, and contained KCs. When the user clicks on a KC, a different window pops up to provide access to the information related to the KC, including relationships, process capability, inspection tolerance, corrective action, and process improvement information. Likewise, when the user clicks on the lines connecting between KCs, a window containing information for the variation propagation relationships appears.

The two-tree modeling methodology provides a way to analyze scenarios where KCs and the product assembly architecture are coupled. When a component or subassembly is scrapped due to one KC not meeting the specification, all KCs on the scrapped component or subassembly are scrapped. The interconnectivity between KCs and components during simulation is further described in Section 4.5.

3.6 Summary

In this Section, we have described the model used by the *KCTool* software. Two model hierarchies are superimposed on top of each other: the product assembly model and the KC variation propagation model. Also, we have demonstrated the process used for deriving the variation propagation transfer coefficients and how the two-hierarchy model is built using the *KCTool* software.

CHAPTER 4: INSPECTION SIMULATION

4.1 Introduction

Once the product assembly model and the KC variation propagation model is created, the cost and benefit of any arbitrary inspection plan is calculated by running a Monte Carlo simulation. Section 4.2 describes Monte Carlo simulations. Section 4.3 describes the random number generator for the normal distributions. Section 4.4 describes the variation propagation and the inspection process. Section 4.6 covers the modeling of rework and scrap. Section 4.7 details how the cost of inspection and the cost of variation are used to calculate the objective cost function at the end of simulation. The last section, Section 4.8 demonstrates the tools using the simple four-part stack-up model introduced in Chapter 2.

4.2 Monte Carlo Simulation

The procedure of generating sample points from the given probability distributions is known as sampling from probability distributions, or random variate generation, or Monte Carlo sampling.

Simulation is a very powerful and widely used management science technique for the analysis and study of complex systems. Simulation may be defined as a technique that imitates the operation of a real-world system as it evolves over time. In contrast to the exact mathematical solutions available with most analytical models, the simulation process involves executing or running the model through time, usually on a computer, to generate representative samples of the performance measures. In this respect, simulation may be seen as a sampling experiment on the real system, with the results being sample points. For example, to obtain the best estimate of the mean of the measure of performance, we average the sample results. Clearly, the more sample points we generate, the better our estimate will be.

As with most other techniques, simulation has its advantages and disadvantages. The major advantage of simulation is that Monte Carlo simulation theory is relatively straightforward. In general, simulation methods are easier to apply than analytical methods. Whereas analytical models may require us to make many simplifying assumptions, simulation models have few such restrictions, thereby allowing much greater flexibility in representing the real system. Once a model is built, it can be used repeatedly to analyze different policies, parameters, or designs. However, it must be emphasized that simulation is not an optimizing technique. It is often used to analyze "what if" types of questions. Optimization with simulation is possible, but it is usually a slow process.

There are two types of simulation models, static and dynamic. A static simulation model is a representation of a system at a particular point in time. We usually refer to a static simulation as a Monte Carlo simulation. A dynamic simulation is a representation of a system as it evolves over time. Within these two classifications, a simulation may be deterministic or stochastic. A deterministic simulation model contains no random variables; a stochastic simulation contains one or more random variables. A discrete simulation model has state variables changing at discrete or countable points, and a continuous simulation model has state variables change continuously. For this research, we use a stochastic Monte Carlo simulation with combinations of discrete and continuous variables.

Monte Carlo simulations provide a way to study processes affected by variation and uncertainty. The sources of variation are modeled as random populations with specified probability distributions. The value for each member of the population is generated randomly and independently according to the specified probability density function. The model is executed with the randomness in the population. A summary of the entire population provides a way to study the expected value of the process outcomes and the amount of variation in the outcome.

Many software applications use Monte Carlo algorithms. An example of a generic system is *Crystal Ball*, which is an add-on for the Excel software, and is often used for financial analysis. Instead of using a distinct value as inputs for the *Excel* formulas, *Crystal Ball* allows users to study the affect of uncertainty on the calculation. An example of a specific Monte Carlo application is the Variation Statistical Analysis (VSA) software, which is often used by industry to model and analyze variation associated with geometric configurations.

4.3 Random Number Generator for Normal Distribution

Each time the simulation process requests a sample point for a KC, a random value is generated according the specified normal distribution. While Visual C++ has a built-in uniform distribution random number generator, a satisfactory normal distribution generator was not found. Therefore, a normal distribution generator algorithm was developed based on the Convolution Algorithm.

In the Convolution Algorithm, we make direct use of the Central Limit Theorem. The Central Limit Theorem states that the sum Y of n independent and identically distributed random variables (Y_1, Y_2, ..., Y_n, each with mean μ and finite variance σ^2) is approximately normally distributed with mean $n\mu$ and variance $n\sigma^2$. If we apply the Central Limit Theorem to uniform-distributed [0,1] random variables, R_1, R_2, ..., R_n, which have mean μ=0.5 and σ^2=1/12, it follows that

$$Z = \frac{\sum_{i=1}^{n} R_i - 0.5n}{\left(\frac{n}{12}\right)^{1/2}} \tag{58}$$

Z is approximately normal with mean 0 and variance 1 (Figure 22). We can expect the approximation to work better as the value of n increases. However, most simulation literature suggests using a value of n=12. Using 12 not only seems adequate but, more important, has the advantage that it simplifies the computational procedure. If we now substitute n=12 into the preceding equation, the process generator simplifies to

$$Z = \sum_{i=1}^{12} R_i - 12 \tag{59}$$

The above equation avoids a square root and a division, both of which are relatively time-consuming routines on a computer. If we want to generate a normal variate X with mean μ and variance σ^2, we first generate Z using the above process and then transform it using the relation $X=\mu+\sigma Z$.

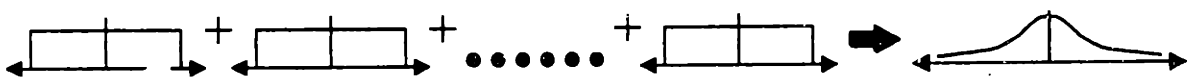


Figure 22: Generating a Gaussian Distribution using Central Limit Theorem

4.4 Variation Propagation and Inspection

In this research, the simulation calculates the cost of production and the quality of the end product for a single product using a back-propagation algorithm. If inspection is specified on a KC, the value of each sample point is inspected after the value is generated. During each sampling of the Monte Carlo simulation, one value is generated for every KC in the flowdown following the four-step process shown in Figure 23.

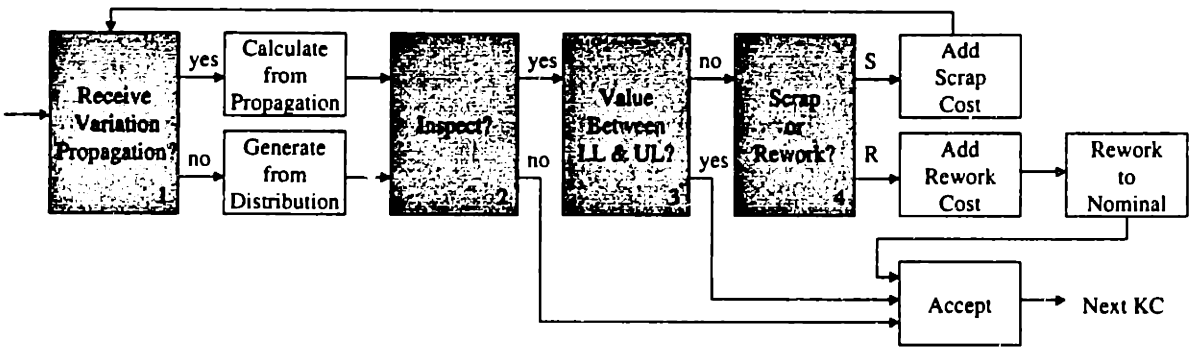


Figure 23: The Inspection Simulation Flow Diagram

In step 1, the program checks if the KC receives variation propagation from other KCs in the KC flowdown (if a value for the contributing KCs has not been generated, the program recursively generates the value). If it does, the value is calculated using the variation propagation matrix described in Section 3.3. Otherwise, the value is randomly generated based on the given variation distribution. In step 2, the program checks if the KC is marked for inspection. If it is, the program goes on to step 3 for inspection. If it is not, the value is accepted. The inspection operation in step 3 checks if the value falls between the specified inspection lower limit (*LL*) and upper limit (*UL*). If yes, the part is accepted. If no, the part is rejected and corrective actions are carried out in step 4. In step 4, either scrap or rework is carried out, until a part is deemed acceptable. The cost and consequences of scrap and rework are discussed in the next section. When the value for every KC is generated, the associated cost is recorded and the process starts over until the inspection simulation produces the number of sample points specified by the user.

4.5 Component-KC Interconnectivity

Since the KCs are features on physical components, the software cannot separate the component assembly model and the KC variation propagation model when running simulations. When the software was developed, extensive attention has been paid to the interconnectivity between the two hierarchy models. Several objectives have to be achieved to accurately account for the way production and inspection is carried out.

1. The product is built one at a time to simulate a pull manufacturing production.

2. Manufacturing starts when the top-level assembly receives a request to be built.

3. When a component is produced, the values for all KCs on the component are set.

4. When a KC request values from its source of variation propagation, it sends a request to all contributing KCs so the components those KCs belong to are produced.

5. When inspection is carried out, any out-of-spec KC value causes the component to be rejected.

6. A rejected component goes through either scrapping the entire component or reworking only the failed KCs to nominal.

7. Scrapping a component causes all KCs in the component to be scrapped and regenerated.

8. When an assembly is scrapped, all components going into the assembly are also scrapped.

The following flowcharts represent the software algorithm used to accomplish the above objectives. Since object oriented programming language C++ is used, each component and each KC is capable of carrying out respective tasks. The component class is responsible for the GetProduct Module, MakeProduct Module, InspectProduct Module, and ScrapProductModule. The KC class is responsible for the GetKC Module, MakeKC Module, InspectKC Module, and ReworkKC Module. The GetContribution Module belongs to the connection class, which is responsible for calculating the variation propagation. More detail explanation of the different modules is in Appendix A.

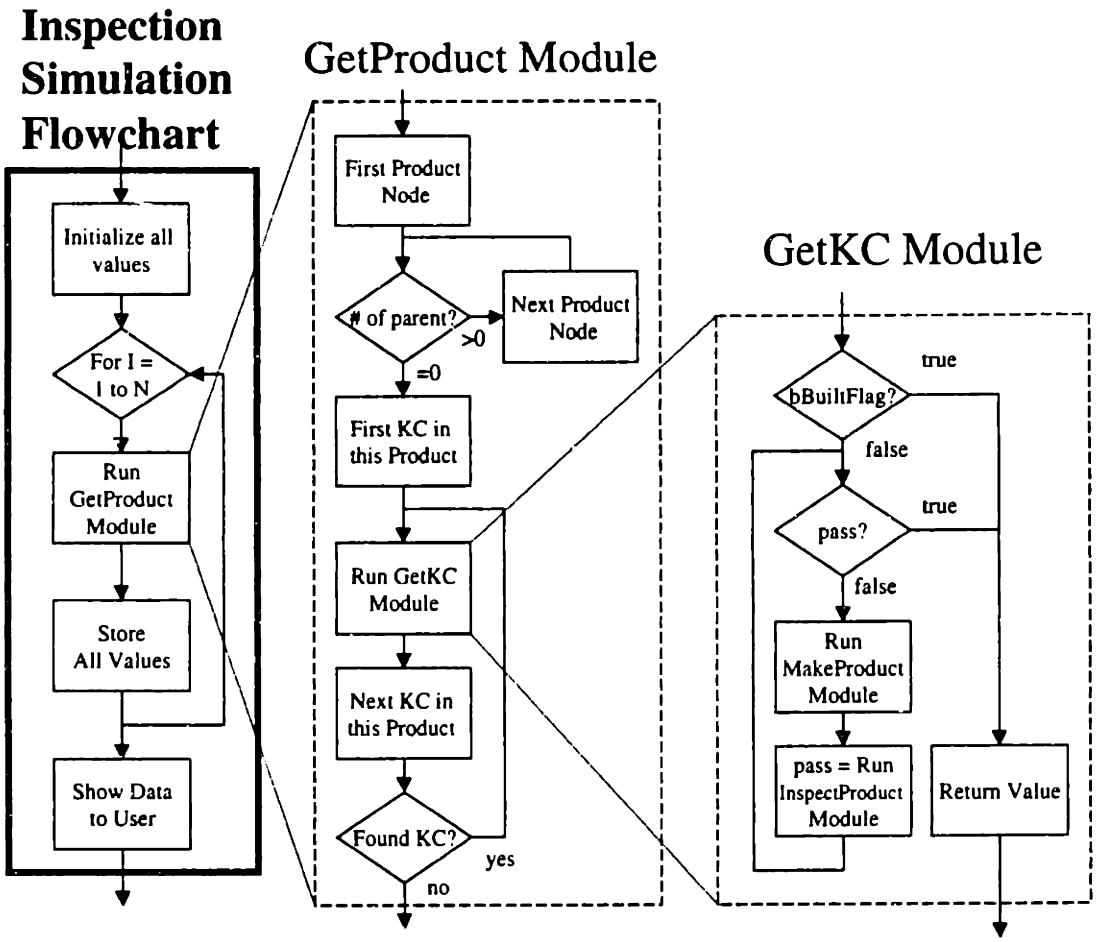


Figure 24: The Simulation Flowchart, the GetProduct Module, and the GetKC Module.

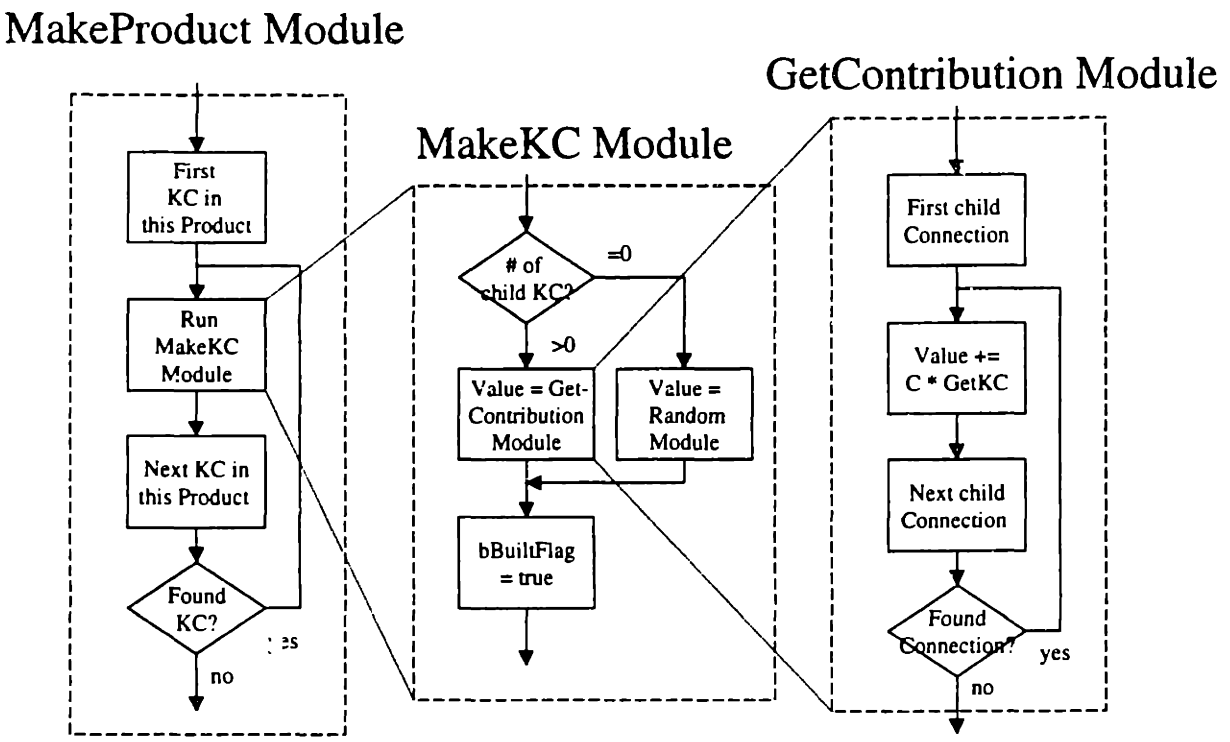


Figure 25: The MakeProduct Module, the MakeKC Module, and the GetContribution Module.

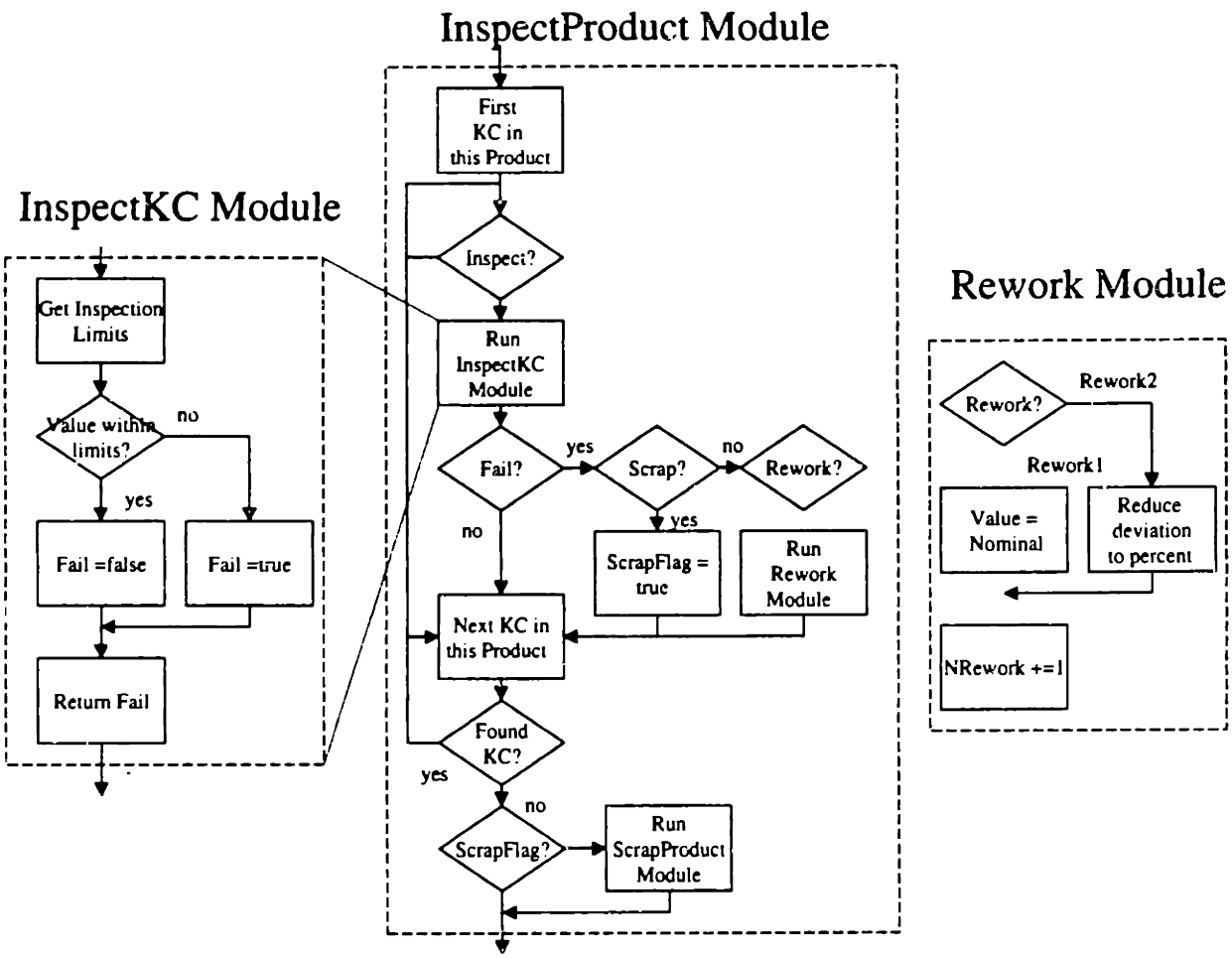


Figure 26: The InspectProduct Module, the InpsectKC Module, and the Rework Module.

4.6 Rework & Scrap Algorithm

When the inspection routine finds value of KC falling outside the inspection specification, two different corrective actions can be taken: rework the failed KC or scrap the entire part. Different cost, benefit, and consequences are associated with each strategy. Reworking a KC directly improves quality by shifting its value closer to the nominal value, as shown in Figure 27. It is assumed that rework is localized and has no effect on other KCs on the same part. The rework cost is specified for each individual KC. When a part or subassembly is reworked, the cost of rework is added to the total cost. In this case, it is assumed that rework brings the value back to nominal. Future extension of the rework function can model cases where it is impossible to bring the out-of-spec values back to nominal since rework can only remove a percentage of the variation. Or certain features can not be isolated for reworking. In this case, instead of only one feature, rework would affect all of the coupled features.

On the other hand, if a part or subassembly is scrapped, all other KC features on the part are also scrapped even if they are within specification. The scrapped part or subassembly is replaced with a new part or new subassembly and inspected again. The process is repeated until an acceptable part is generated (Figure 28). When a part is scraped the cost of the new part or new subassembly is added to the total cost.

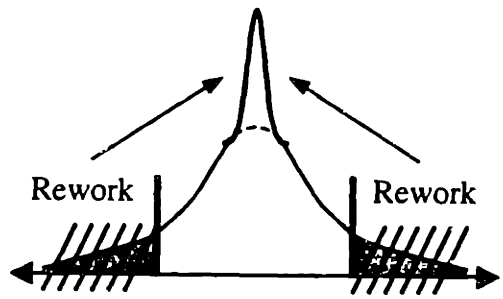


Figure 27: Effect of Rework

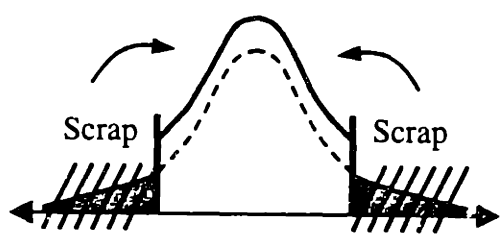


Figure 28: Effect of Scrap

4.7 Cost Accounting Algorithm

Inspection Cost. Besides the cost associated with scrapping a part, C_{scrap}, and reworking a feature, C_{rework}, inspection itself also has an associated cost. Implementing an inspection incurs both a fixed cost and variable cost. The fixed cost is the cost of installing inspection stations and apparatus and the variable cost is the recurring labor cost, C_i.

Inspection Benefit. The goal of inspection is to improve product quality (reduce the number of parts falling outside of specification) by inspecting and reworking or scrapping parts before assembly takes place. To quantify the benefit, either a step function or the Taguchi Loss Function as shown in Figure 29 can be used to quantify the reduced quality associated with variation.

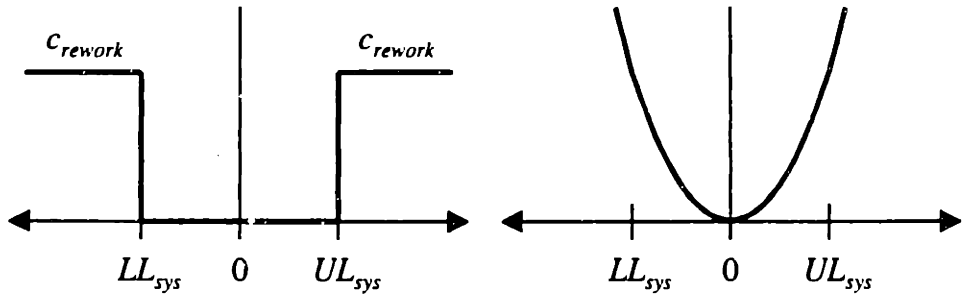


Figure 29: Step Function and Taguchi Lost Function

While Taguchi Loss Function takes the form of Equations 53 and captures the cost of minor variation, the step function captures the way costs are triggered in industry more accurately [Thornton 1997] and has numerous industry cost data. Although Taguchi Lost Function is a widely accepted concept to measure quality, the exact function is rarely used.

$$c_{lost} = k(\Delta y)^2 \tag{60}$$

The simulation in this thesis utilizes the step function approach. Whenever a product-KC falls outside the specification, the cost of reworking the product feature is added to the product failure cost, $C_{failure}$. Due to the assumption that all product-KCs are inspected and reworked if unacceptable, the user is not able to add extra inspection for all product-KCs. Besides the Taguchi Loss Function and the step function, a modified Taguchi Loss Function can also be used. It is formulated as

$$C_{failure} = c_{rework}\left(\frac{\Delta y}{UL_{sys}}\right)^2 \text{ if } 0 < \Delta y < UL_{sys} \tag{61}$$

$$C_{failure} = c_{rework}\left(\frac{\Delta y}{LL_{sys}}\right)^2 \text{ if } LL_{sys} < \Delta y < 0 \tag{62}$$

$$C_{failure} = c_{rework} \text{ if } \Delta y < LL_{sys} \text{ or } \Delta y > UL_{sys} \tag{63}$$

The above equations describe the cost structure as shown in Figure 30. It captures both the step cost of performing rework to eliminate variation outside of specification tolerance and the quality lost cost associated with minor variation within the specification. Whether this is a valid cost accounting methodology has not yet been determined.

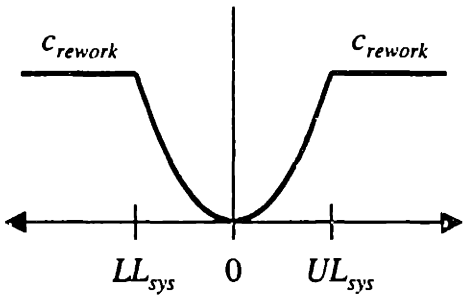


Figure 30: Modified Taguchi Cost Function

The total cost, C_T, is the sum of unit production cost, inspection cost, scrap cost, rework cost, and failure cost. C_T forms the objective function for the simulation.

$$C_T = C_m + C_{inspection} + C_{scrap} + C_{rework} + C_{failure} \tag{64}$$

4.8 Verification

While analysis for the simple assembly stack-up in Chapter 2 can be used to explicitly find the optimum inspection tolerances, the model was tested in *KCTool* to confirm the two approaches.

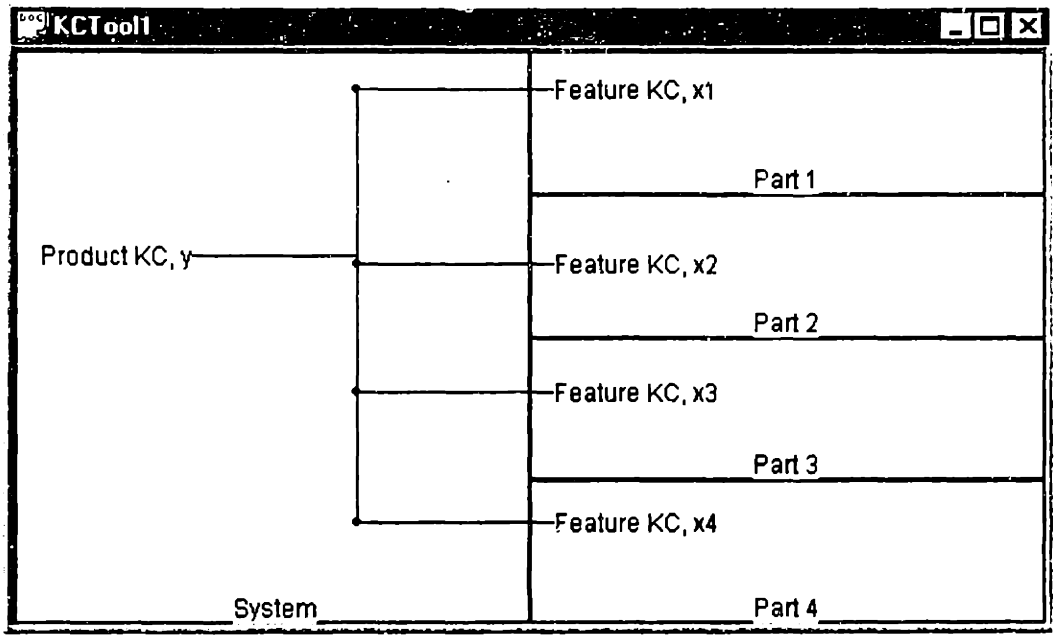


Figure 31: Product Decomposition and KC Flowdown for Simple Assembly Stack-Up

The simulation was run with the parameter values from Table 1. Figure 31 shows the flowdown model in *KCTool*. 10,000 sample points were generated for each value of t_j. Figure 32 shows the plot for both the

analysis and the simulation. The two methods yield consistent results and have a variance of 0.0061. The variance is caused by the random number generator not generating perfect normal distributions. Increasing the number of simulation sample points alleviates the problem.

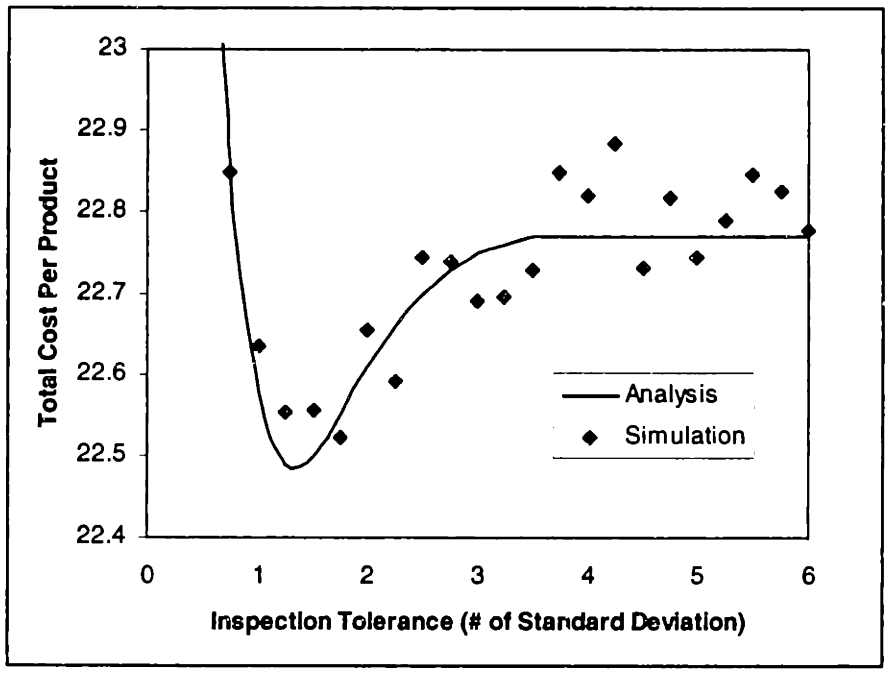


Figure 32: Total Cost Calculations for Analysis and Simulation

4.9 Summary

Using the model described in Chapter 3, we have now demonstrated how Monte Carlo simulation can be implemented to analyze an inspection problem. Running the simulations using different inspection allocation, inspection tolerance settings, and corrective actions, allow us to compare the cost of inspection and the cost saving from the quality improvement for the different inspection plans. Using the simulations, engineers can test the different inspection plans during the design phase of the product development, rather than later in the production phase.

CHAPTER 5: INSPECTION OPTIMIZATION

5.1 Introduction

Chapter 4 demonstrated how the cost and the benefit of an inspection plan can be calculated for a specific inspection plan. As shown in the example, it is possible for a design team to test various scenarios and select the best. However, in a complex system, testing different combinations of inspection locations, inspection limits, and rework vs. scrap options is very time consuming. To help the design team identify the optimal plan, an computer-based optimization method was applied. Section 5.2 describes the different optimization methodologies, Section 5.3 details the algorithm used in this research, Section 5.4 introduces a case study of the aircraft wing contour and the simulation results, and Section 5.5 discusses factors affecting the optimization performance.

5.2 Optimization Methodologies

Traditional optimization problems often employ one of many stochastic optimization techniques. They include conjugate gradient optimization, linear and integer programming, taboo search, genetic algorithm, and simulated annealing.

5.2.1 Conjugate Gradient Optimization

The Conjugate Gradient Optimization is a minimization technique to find the lowest energy. What it does is similar to a steepest descent minimization. The procedure is as follows: firstly the gradient is calculated in all directions. The path of steepest descent is chosen. This is in the form of a straight line but making measurements of the energy at small increments. When the energy stops decreasing the process is repeated until the minimum energy is found. Conjugate Gradient minimization adopts the same procedure, but also uses information from the previous iteration steps to select the optimal route. Disadvantage of the Conjugate Gradient Optimization technique is that it can only be applied to continuous solution space.

5.2.2 Linear and Integer Programming

Linear and integer programming have proved valuable for modeling many and diverse types of problems in planning, routing, scheduling, assignment, and design. Industries that make use of LP and its extensions include transportation, energy, telecommunications, and manufacturing of many kinds.

A *Linear Program* (LP) is a problem that can be expressed as follows (the so-called Standard Form):

```
minimize cx
subject to Ax=b
x >= 0
```

where x is the vector of variables to be solved for, A is a matrix of known coefficients, and c and b are vectors of known coefficients. The expression "cx" is called the objective function, and the equations "$Ax=b$" are called the constraints. All these entities must have consistent dimensions, of course, and you can add "transpose" symbols to taste. The matrix A is generally not square, hence you don't solve an LP by just inverting A. Usually A has more columns than rows, and $Ax=b$ is therefore quite likely to be under-determined, leaving great latitude in the choice of x with which to minimize cx.

Although all linear programs can be put into the Standard Form, in practice it may not be necessary to do so. For example, although the Standard Form requires all variables to be non-negative, most good LP software allows general bounds $l <= x <= u$, where l and u are vectors of known lower and upper bounds. Individual elements of these bounds vectors can even be infinity and/or minus-infinity. This allows a variable to be without an explicit upper or lower bound, although of course the constraints in the A-matrix will need to put implied limits on the variable or else the problem may have no finite solution. Similarly, good software allows $b1 <= Ax <= b2$ for arbitrary $b1$, $b2$; the user need not hide inequality constraints by the inclusion of explicit "slack" variables, nor write Ax >= b1 and $Ax <= b2$ as two separate constraints. Also, LP software can handle maximization problems just as easily as minimization (in effect, the vector c is just multiplied by -1).

The importance of linear programming derives in part from its many applications and in part from the existence of good *general-purpose* techniques for finding optimal solutions. These techniques take as input only an LP in the above Standard Form, and determine a solution without reference to any information concerning the LP's origins or special structure. They are fast and reliable over a substantial range of problem sizes and applications.

Two families of solution techniques are in wide use today. Both visit a progressively improving series of trial solutions, until a solution is reached that satisfies the conditions for an optimum. *Simplex* methods

visit "basic" solutions computed by fixing enough of the variables at their bounds to reduce the constraints $Ax = b$ to a square system, which can be solved for unique values of the remaining variables. Basic solutions represent extreme boundary points of the feasible region defined by $Ax = b$, $x >= 0$, and the simplex method can be viewed as moving from one such point to another along the edges of the boundary. *Barrier* or *interior-point* methods, by contrast, visit points within the interior of the feasible region. These methods derive from techniques for nonlinear programming that were developed and popularized in the 1960s by Fiacco and McCormick, but their application to linear programming dates back only to Karmarkar's innovative analysis in 1984.

The related problem of integer programming (or integer linear programming, strictly speaking) requires some or all of the variables to take integer (whole number) values. Integer programs (IPs) often have the advantage of being more realistic than LPs, but the disadvantage of being much harder to solve. The most widely used general-purpose techniques for solving IPs use the solutions to a series of LPs to manage the search for integer solutions and to prove optimality.

5.2.3 Taboo Search

Taboo search is an iterative procedure for solving discrete combinatorial optimization problems. It was first suggested by Glover [1977] and since then has become increasingly used. It has been successfully applied to obtain optimal or sub-optimal solutions to such problems as scheduling, time-tabling, travelling sales person, and layout optimization. The method, described by Glover, Taillard and de Werra [1993], is to explore the search space of all feasible solutions by a sequence of moves. A move from one solution to another is the best available. However, to escape from locally optimal but not globally optimal solutions and to prevent cycling, some moves, at one particular iteration, are classified as forbidden or *taboo*. Taboo moves are based on the short-term and long-term history of the sequence of moves. A simple implementation, for example, might classify a move is taboo if the reverse move has been made recently or frequently. Sometimes, when it is deemed favorable, a taboo move can be overridden. Such aspiration criteria might include the case which, by forgetting a move is taboo, leads to a solution which is the best obtained so far.

5.2.4 Genetic Algorithm

Genetic algorithms were invented by Holland [1975] to mimic some of the processes of natural evolution and selection. In nature, each species need to adapt to a complicated and changing environment in order to maximize the likelihood of its survival. The knowledge gained by each specie is encoded in its

chromosomes which undergo transformations when reproduction occurs. Over a period of time, these changes to the chromosomes give rise to species that are more likely to survive, and so have a greater chance of passing their improved characteristics on to future generations. Of course, not all changes will be beneficial but those which are not tend to die out.

Holland's genetic algorithm attempts to simulate nature's genetic algorithm in the following manner. The first step is to represent the different components of a solution as different continuous or discrete variables. Then an initial population of solutions is constructed at random. At each generation, the fitness of each solution in the population is measured (a high fitness value would indicate a better solution than a low fitness value). The better solutions are then selected to produce offspring for the next generation, which inherit the best characteristics of both parents. After many generations of selection for the fitter solutions, the result is hopefully a population that is substantially fitter than the original. The theoretical basis for the genetic algorithm is the *Schemata Theorem* [Holland 1975], which states that the individual solutions with good, short, low-order schemata or building blocks (i.e. beneficial parts of the solution) receive an exponentially increasing number of trials in successive generations.

5.2.5 Simulated Annealing

Simulated annealing (SA) is a stochastic computational technique derived from statistical mechanics for finding near globally-minimum-cost solutions to large optimization problems. In many instances, finding the global minimum value of an objective function with many degrees of freedom subject to conflicting constraints is a NP-complete problem, since the objective function will tend to have many local minimums. A procedure for solving optimization problems of this type should sample the search space in such a way that it has a high probability of finding the optimal or a near-optimal solution in a reasonable time. Over the past decade or so, simulated annealing has shown itself to be a technique which meets these criteria for a wide variety of applications.

Kirkpatrick *et al.* [1983] were the first to propose and demonstrate the application of simulated annealing techniques to problems of combinatorial optimization. The simulated annealing algorithm models after the metal annealing process. It starts at high temperature, T_{start}, and is slowly cooled to the ending temperature, T_{end}. Besides keeping track the current settings and the current objective function result, the process also keeps track the best objective function result and its input settings. During each increment of the cooling process, one input variable randomly selected from all variables is changed, and this results in a movement into a neighboring solution point. Since the variables are selected randomly, the process works with combinations of discrete and continuous variables. Discrete variables would change to

different discrete values, while continuous variables would change at random magnitudes. After a new solution settings are set, a new objective function calculation is performed using the new settings. If the new objective function is better than the previous, the best objective function and the best solution is replaced by the new settings. From the new solution point, subsequent movement is generated. In addition to replacing the best when a better solution is found, the simulated annealing also allows movement to a worse solution point. Just like molecules having high kinetic energy at high temperature, the simulated annealing process gave the transition process a high probability for moving toward an inferior position initially, in hope of eventually moving past the inferior solutions and finding the global optimum. The probability of accepting an inferior solution depends on both the temperature of the state, T, and the difference in objective functions, ΔC_T:

$$p_{accept} = e^{\frac{\Delta C_T}{T}} \tag{65}$$

Since T decreases overtime as $T_{k+1}=\alpha T_k$ and α is generally between 0.8 to 0.99, the probability of accepting an inferior solution decreases overtime. Also, large ΔC_T makes the probability lower as well, as the simulated annealing process allows small declines in optimality but discourage large declines. The process repeats until T_{end} is reached.

5.2.6 Comparisons

For continuous functions, gradient decent algorithms can help find the maximum or minimum solutions in the continuous solution space. For problems with linear objective functions and linear constraint conditions, Linear programming can be used to find the optimal solution, which lies at one of the vertexes of the boundary. For problems with discrete solution spaces, integer programming and dynamic programming algorithms are often used.

Traditional optimization algorithms are not appropriate for this optimization problem because the search space contains a mix of discrete and continuous variables. Also, many local minimums exist in the search space. In addition, the objective function for inspection is not linear. All these make it difficult for traditional optimization algorithms to find a global minimum.

Stochastic search techniques, such as genetic algorithms or simulated annealing, can be applied in the inspection problem because of their ability to robustly find the global minimum in mixed discrete and continuous spaces with multiple local minimums. The optimization algorithm for this research was

formulated using simulated annealing because of its simplicity and low computational overhead as demonstrated by Thornton and Johnson [1996] and Szykman and Cagan [1997].

5.3 Simulated Annealing Search Algorithm

The simulated annealing algorithm is applied to find the optimal inspection plan. The overall optimization process is shown in Figure 33. The optimization routine performs many simulations with different inspection settings selected by the simulated annealing process. Three arrays of size n (the total number of KCs from every layer of KC flowdown) are used to store the inspection settings for the whole system.[6] The first array is a binary valued vector $S=\{s_1, s_2...s_n\}$ where s_i is 1 if the particular KC is inspected otherwise it is 0. The second array is a continuous valued vector $T=\{t_1, t_2...t_n\}$ where t_i represents inspection tolerance tightness. The third array is a binary valued vector $A=\{a_1, a_2...a_n\}$ where a_i is 0 if the part containing the non-conforming feature is scrapped when it fails inspection otherwise a_i is 1 if the non-conforming feature is reworked. The value for the third array is only relevant when a feature is inspected.

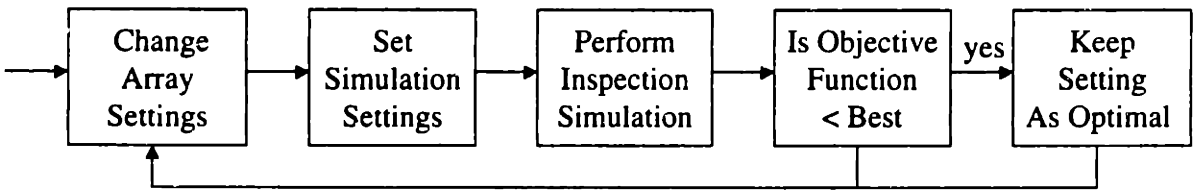


Figure 33: Optimization Flow Diagram

At the beginning of the optimization process (T_{start}), the Simulated Annealing algorithm randomly sets the three arrays. The transition between subsequent array configuration is done by randomly selecting a feature, i, and changing s_i to 1 if s_i=0 or randomly picking s_i, t_i, or a_i, to change if s_i=1. After setting the inspection specification, an inspection simulation is performed and total cost is calculated. If the cost of the new plan is lower, the new configuration is accepted. However, if the new objective function is higher, then the transition to the new configuration is accepted with a specified probability. The probability is calculated using Equation 65. The process repeats until the stopping criterion T_{end} is reached.

[6] The user has the option to turn off inspection of certain KCs.

5.4 Case Study: Aircraft Wing Contour

We will now use a case study from the aircraft industry to demonstrate the optimization function.

An aircraft wing has three main Sections (shown in Figure 34), the center box, leading edge, and trailing edge. The center box, the focus of this case, provides most of the wing structural stiffness, surface area, and attach point to the fuselage box. It has several variation sensitive requirements including the steps and gaps between the skin panels, the orientation of the wing, and the contour of the upper and lower surfaces.

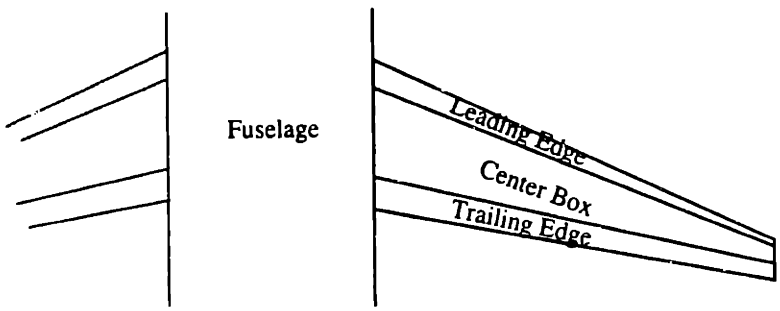


Figure 34: Wing Configuration.

The case study will focus on the contour of center box surface. The center box is comprised of three main structures: spars, skins, and ribs. The assembly tree is shown in Figure 35. The spars are built up from two T-shaped extrusions called cords and a web, usually a reinforced aluminum sheet. These are assembled separately in their own fixtures and then the entire assembly is built in the wing fixture. The spars are placed in the fixture first. The ribs are located relative to the spars using the locating holes in the web. Finally, the skins are attached to the top and bottom surfaces defined by the ribs and spars.

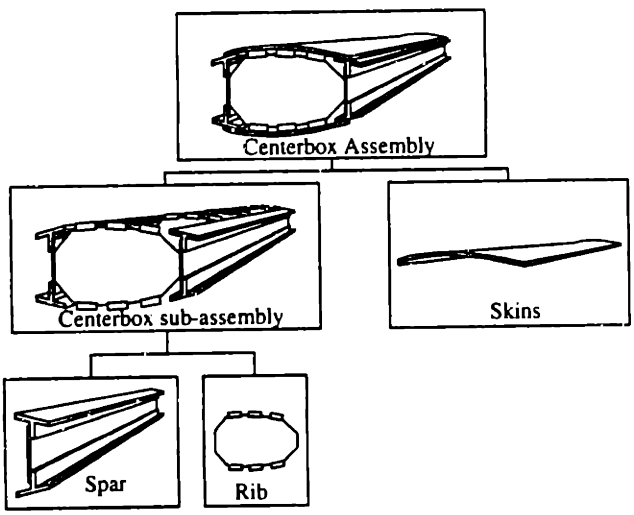


Figure 35: Centerbox Assembly

The case study focuses on two system-KCs for the wing: leading edge and middle contours. These dimensions are the relative heights of the two contour points measured from aircraft coordinates. These product-KCs are flowed down to the sub-assembly features by analyzing the tolerance stack-ups. The tolerance paths are shown in Figure 36.

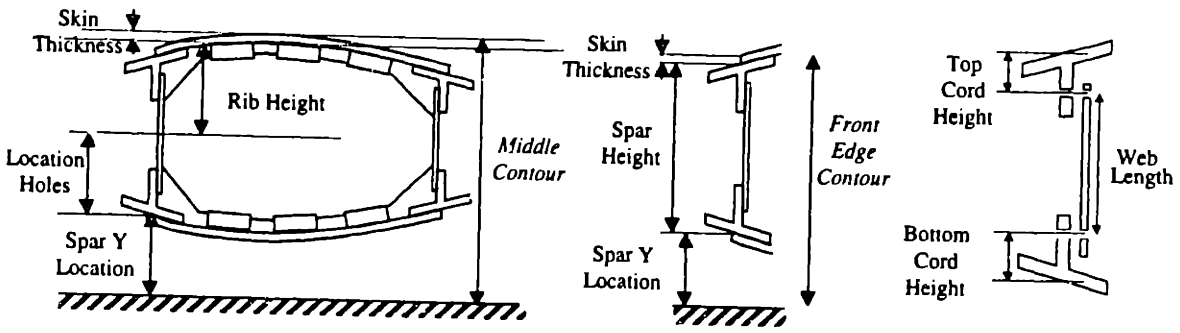


Figure 36: Tolerance Stack-Ups for Leading Edge and Middle Contour

The spar Y location is the error in the location fixture. The skin thickness and location holes are machined features. The rib height and spar heights can be flowed down further based on their assembly process. The ribs are assembled in a fixture: the parts are located relative to each other in a fixture and match drilled. The spars are assembled using a determinate assembly process: the parts are located relative to each other using precision location holes. The above KC flowdown information is converted to three variation propagation equations:

$$\textit{Middle Contour} = \textit{Spar Y Location} + \textit{Location Holes} + \textit{Rib Height} + \textit{Skin Thickness} \quad (66)$$

$$\textit{Front Edge Contour} = \textit{Spar Y Location} + \textit{Spar Height} + \textit{Skin Thickness} \quad (67)$$

$$\textit{Spar Height} = \textit{Top Chord Height} + \textit{Web Length} + \textit{Bottom Chord Height} \quad (68)$$

Figure 37 shows a screen dump from the *KCTool*'s model. It captures both the product decomposition and the KC variation propagation. While most features contribute to variation at either the front edge contour or the middle contour, the spar Y location of the centerbox subassembly and skin contribute to both. Also, while most parts and subassemblies have only one KC per physical part in this example, both the centerbox assembly and the spar have two KCs per assembly.

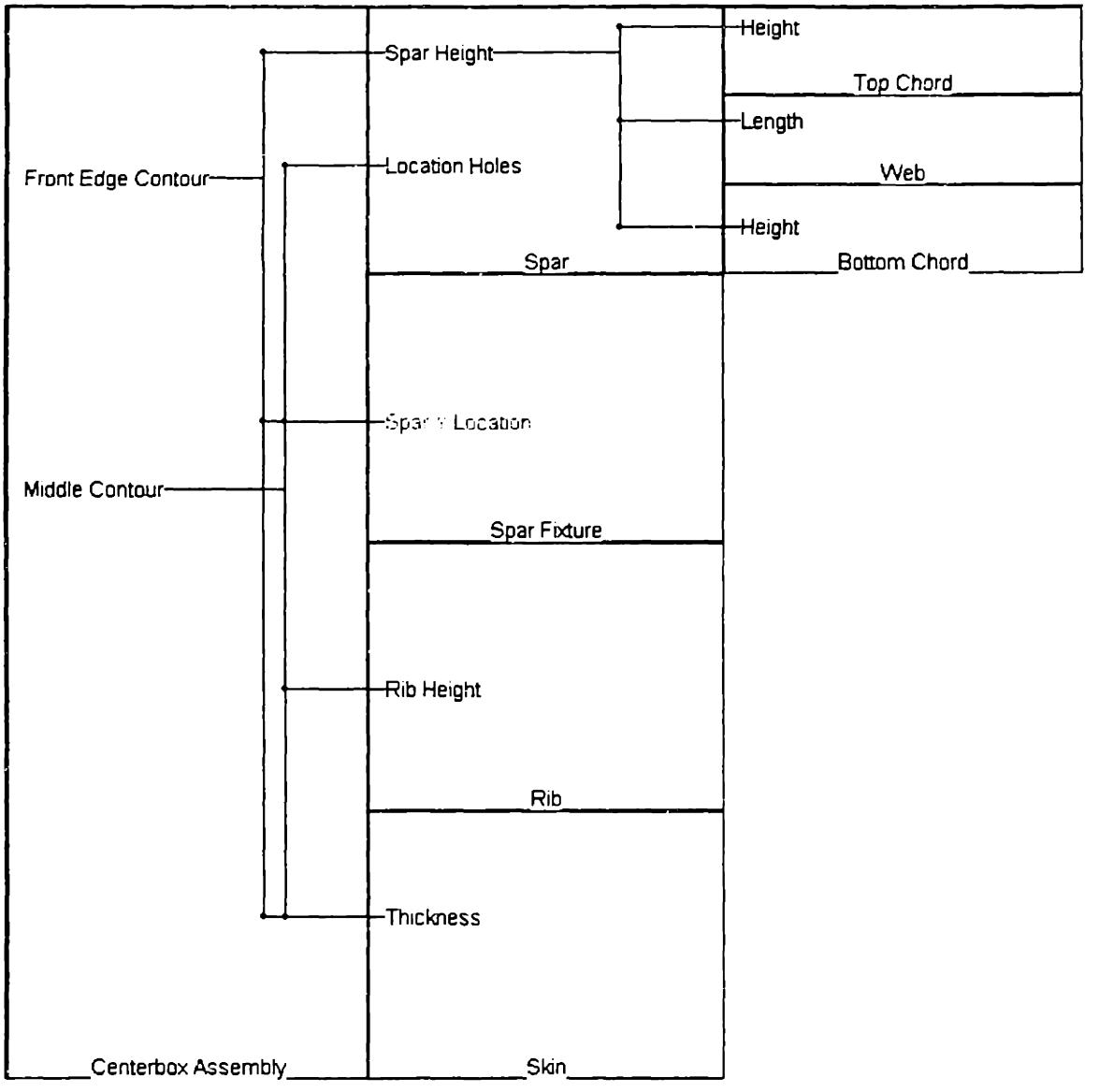


Figure 37: Centerbox Assembly *KCTool* Model

Table 2 shows the process capability and cost information used for the analysis. The numbers have been modified to mask any proprietary data. Scrap cost represents the cost of throwing away a part, subassembly, or entire assembly. It includes the part costs and the assembly cost. The rework cost represents the cost of reworking an out-of-spec feature to nominal. The table shows seven sources of manufacturing and assembly variation: top chord height, web length, bottom chord height, rib height, spar Y location, location holes, and skin thickness. The front edge contour and the middle contour of the centerbox assembly are assumed to have *UL* and *LL* of –0.02 and +0.02. All the features are assumed to have $C_{inspection}$ of $20.

Table 2: Costs and process capability of the center box assembly

Assembly	Assembly Cost	Scrap Cost	KCs	$\mu_i - m_i$	σ_i	Rework Cost
Centerbox Assembly	$4500	$8000	Front Edge Contour	-	-	$3000
			Middle Contour	-	-	$3000
Spar	$800	$2000	Spar Height	-	-	$700
			Location Holes	0	0.005	$500
Top Chord	-	$300	Height	0	0.008	$100
Web	-	$600	Length	0	0.005	$100
Bottom Chord	-	$300	Height	0	0.008	$100
Spar Fixture[7]	-	-	Spar Y Location	+0.003	0.002	-
Rib	-	$1000	Height	-0.005	0.009	$200
Skins	-	$500	Thickness	0	0.005	$100

- => mean shift and σ determined by propagation from its contributing sources of variation

From the simulation, when only the product-KCs are inspected and reworked (i.e., no inspection throughout the assembly process), the total cost is $9301 per unit (i.e., extra $1301 is spent on reworking out-of-spec product-KCs).

When running the optimization routine, each simulation was run for 1,000 sample points and the simulated annealing parameters were set at T_{start}=1000, T_{end}=0. 1, and α=0. 99. The optimization took 3

[7] The variation in the spar fixture is assumed to be unmeasurable and therefore excluded from the inspection plan.

hour and 36 minutes[8] on a Pentium 120MHz PC and the result is shown by the *KCTool* screen dump in Figure 38. The shaded KCs indicate inspection. The inspection limits shown underneath are measured as the number of standard deviation away from nominal, t_i. Of those inspected KCs, the best corrective actions, whether to scrap nonconforming parts or rework nonconforming features, are shown as well. When the optimization was run with the same settings but doubling the number of simulations for each value of T, the same inspection points were selected but the limits on the points were different slightly.

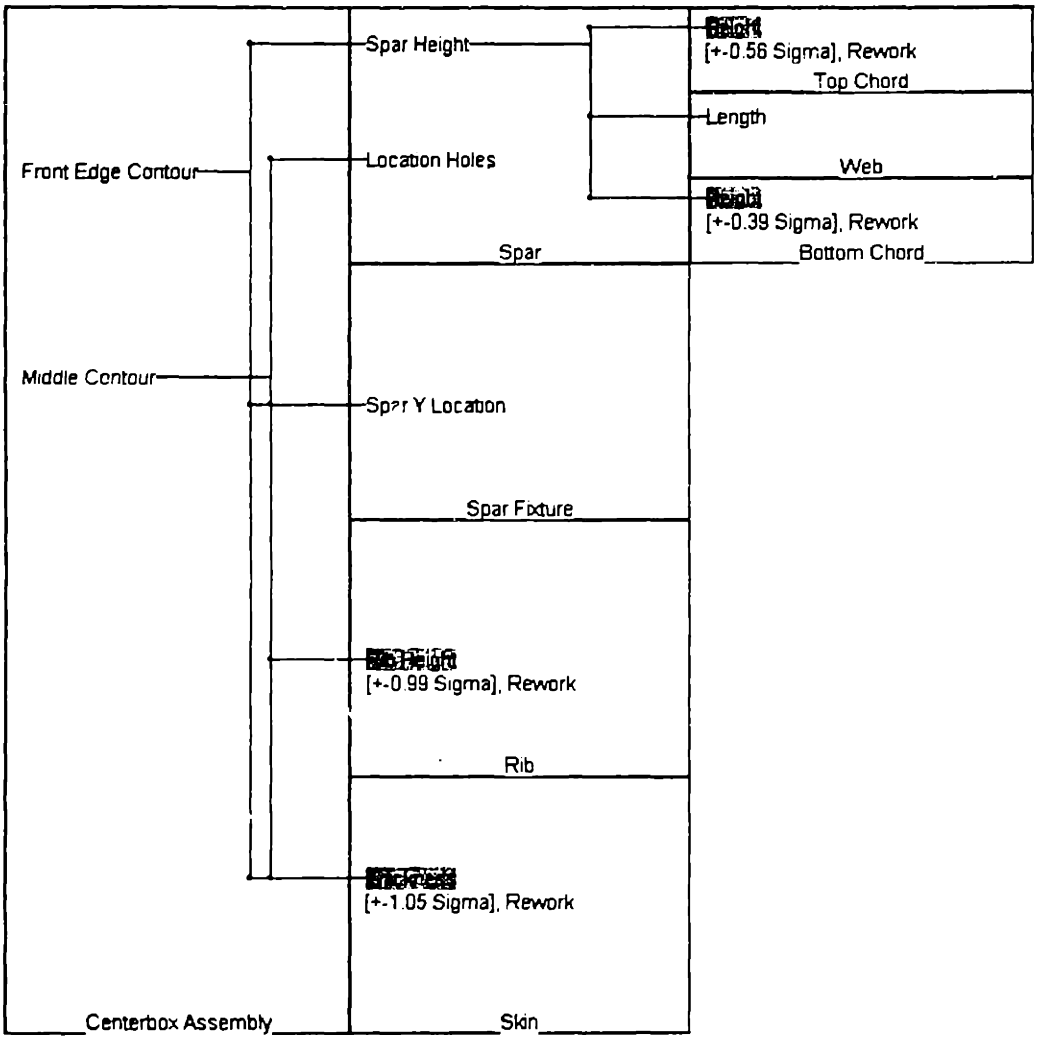


Figure 38: An Optimal Inspection Plan Selected by the Optimization Algorithm.

The solution shows a combination of reworking top chord height, bottom chord height, rib height and skin thickness to be the best inspection plan. Performing inspection simulation with the given solution

[8] The random number generator used n=1000 instead of n=12.

yields a unit cost of $8874 ($874 in excess quality costs). This translates into a reduction of 32.8% in extra cost.

The solution makes intuitive sense because the top chord and bottom chord of the spar and the height of the rib contribute more variation than other parts and have a low cost of rework. The skin's contribution of variation is the same as the spar location holes and the web length, but it has a lower rework cost.

One issue should be noted in the optimization result. It shows different inspection tolerance for the heights of the top and bottom chords even though they have the same process capability and contribute the same variation to the front edge contour. Also, different number of sample points for each simulation results in same inspection plan but different inspection limits. The difference is caused by the simulated annealing process, which has a hard time finding the absolute optimal in a continuous solution space, but rather only finds within a range.

As a result, the following specifications are chosen:

Table 3: Final optimal inspection plan

KCs	Inspection LL and UL	Corrective Action
Height of top chord	[-0. 004, +0. 004]	Rework
Height of bottom chord	[-0. 004, +0. 004]	Rework
Rib height	[-0. 009, +0. 009]	Rework
Skin thickness	[-0. 005, +0. 005]	Rework

Running a simulation with the above specification results in a unit cost of $8874 (same result as the optimization solution). The result shows that the simulation is insensitive to minor changes in inspection simulation.

5.5 Factors Affecting the Performance of Optimization

Due to the limited computational time, several factors affect the performance, accuracy, and efficiency of the simulated annealing optimization.

1. Production Lot Size. The more data points each simulation uses, the closer Monte Carlo simulation reflect the process capability, and the more accurate the simulation is. However, there is a proportional increase in running time. For the wing contour simulation, each zero-inspection simulation with 1,000 data points takes an average of 6 seconds, and the variability error is 1.2% (the variance for 20 identical simulations). When the number of data points for each simulation is increased 10-fold, the running time increases 10-fold to about 57 seconds, but the variability drops to 0.3%.

2. Simulated Annealing Parameters. The T_{start}, T_{end}, and α parameters influence the number of simulation runs and the minimum found. Using T_{start}=1000, T_{end}=0.1, and α=0.9, the process runs 176 simulations, and the solution yields a reduction of 15.7% over the no-inspection baseline case. Increasing α to 0.99 increases the number of simulations to 1834 but yields an inspection plan resulting in a 32.8% reduction in extra cost.

Table 4: Result under different simulated annealing parameters

	Reference	Case 1	Case 2	Final Plan
α	-	0.9	0.99	-
# of simulations	-	176	1834	-
Running Time	-	21.5 minutes	216.0 minutes	-
Excess costs	$1301	$1100	$874	$874
Percent Saving	-	15.7%	32.8%	32.8%

3. Array-Changing Algorithm. There are several methods to generate transitions between search points. Selecting the best approach is difficult for two reasons. First, the model has a large number of possible solutions (at least n^3 if only choosing between no inspection, inspection then rework, and inspection then scrap, which makes 177,147 combinations for the wing contour case). There is a tradeoff between testing sufficiently diverse solutions and fine-tuning the solution. Second, the model is formulated such that different values for array T and array A result in different inspection settings only when the value for array S is 1. As a result, the algorithms for changing the three arrays need to

be set in a fashion balancing the similarity of consecutive simulations and the elimination of identical settings (not running simulations representing identical inspection plans).

4. Normal Distribution Algorithm. The current algorithm for the normal distribution random number generator is based on the Central Limit Theorem. However, the current software was developed summing 1000 uniform-distribution random variables to build one value instead of 12 suggested by other literatures. Further improvement to the software can use the lesser number to generate the normal distribution to make the simulation less computation-intensive without compromising the integrity of the normal distributions.

5.6 Summary

The optimization process has shown to be an effective way of finding the optimal inspection settings for the complex assembly and variation propagation problem. While many different optimization methods are available, the simulated annealing optimization method is an appropriate method to perform optimization in a combination of discrete and continuous solution space. The aircraft wing contour example demonstrate that the optimization process can be applied to find the optimal inspection plan for the wing contour and achieve the highest quality improvement at the lowest cost.

CHAPTER 6: CONCLUSIONS

In conclusion, this thesis has demonstrated that an optimal inspection plan can be quantitatively selected using the KC flowdown model and Monte Carlo simulation. On one had, this research laid the foundation for formulating analytical equations to evaluate inspection plans; on the other hand, the research has provided a new methodology to help select the optimal inspection plan based on modeling, simulation, and optimization. While other researchers have explored various aspects of inspection optimization for serial and non-serial multistage production process and multi-characteristic products, this is the first research combining the two aspects and implementing the algorithms into a software tool. The KC flowdown model provides a way to understand, visualize, and capture the way variation propagates from feature-level KCs to system requirements through multiple-level cause-and-effect relationships. The Monte Carlo simulation calculates the cost and benefit of an inspection plan. Running multiple simulations in an optimization process provides a way to find an optimal inspection plan.

Several issues require further research. A better transition strategy needs to be developed. In addition, the possibility of mixing simulated annealing with gradient decent algorithms will be explored. To improve the usability, additional methods to increase the efficiency of the software should also be implemented. Possible future extensions include relaxing the bilateral tolerance assumption, modeling the uncertainty in rework effectiveness, and testing non-unity transfer coefficients. While variation reduction optimization has been an area already quantified by [Thornton 1998], other variation risk mitigation approaches, such as process change and design change, will also be developed in the long term. Combining a variety of optimization algorithms will allow companies to determine, quantitatively, the most cost-effective approach for variation risk mitigation.

REFERENCES

Ardayfio, M. A. (1998). Methods for Capturing Design Intent Using Key Characteristics. Mechanical Engineering. Cambridge, MA, MIT.

Bisgaard, S., W. G. Hunter, et al. (1984). "Economic Selection of Quality of Manufactured Product." *Technometrics* 26: 9-18.

Carlsson, O. (1989). "Economic Selection of a Process Level under Acceptance Sampling by Variables." *Engineering Costs and Production Economics* 16: 69-78.

Chakravarty, A. K. and A. Shtub (1987). "Strategic Allocation of Inspection Effort in a Serial, Multi-Product Production System." *IIE Transactions* 19(1): 13-22.

Chen, S.-L. and K.-J. Chung (1996). "Selection of the Optimal Precision Level and Target Value for a Production Process: The Lower-Specification-Limit Case." *IIE Transactions* 28: 979-985.

Clifford, H. (1997). Six Sigma. Continental: 64-67.

Cunningham, T. W., R. Mantripragada, et al. (1996). *Definition, Analysis and Planning of a Flexible Assembly Process*. Proceedings of the Japan/USA Symposium on Flexible Automation, Boston, MA.

Doherty, J. (1997). Mixed Signals: After a Search, AlliedSignal's Chief Aims to Win the Street's Favor Again. Barrons: 29-34.

Drake, A. W. (1967). *Fundamentals of Applied Probability Theory*. New York, McGraw-Hills Book Company.

Elmaghraby, S. E. (1986). "Comments on a DP Model for the Optimal Inspection Strategy." *IIE Transactions* March: 104-108.

Frey, D., K. Otto, et al. (1998). "Evaluating Process Capability Given Multiple Acceptance Criteria." *Journal of Manufacturing Science and Engineering*.

Frey, D. D., K. N. Otto, et al. (1997). "Swept Envelopes of Cutting Tools in Integrated Machine and Workpiece Error Budgeting." *CIRP Annals - Manufacturing Technology* 46(1): 475-480.

Gao, J., K. W. Chase, et al. (1998). "Generalized 3-D Tolerance Analysis of Mechanical Assemblies with Small Kinematic Adjustments." *IIE Transactions* 30: 367-377.

Glover, F. (1977). "Heuristics for Integer Programming using Surrogate Constraints." *Decision Sciences* 8: 156-166.

Glover, F., T. E., et al. (1993). "A User's Guide to Tabu Search." *Annals of Operations Research* 41: 3-28.

Greenshtein, E. and G. Rabinowitz (1997). "Double-Stage Inspection for Screening Multi-Characteristic Items." *IIE Transactions* 29: 1057-1061.

Holland, J. H. (1975). *Adaptation in Natural and Artifical Systems*. Ann Arbor, The University of Michigan Press.

Homann, B. S. and A. C. Thornton (1998). "Precision Machine Design Assistant: A Constraint-Based Tool for the Design and Evaluation of Precision Machine Tool Concepts." *AIEDAM* 12: 419-429.

Hunter, W. G. and C. P. Kartha (1977). "Determining the Most Profitable Target Value for a Production Process." *Journal of Quality Technology* 9: 16-25.

Kirkpatrick, S., C. D. G. Jr., et al. (1983). "Optimization by simulated annealing." *Science*: 671-680.

Lee, D. and A. Thornton (1996). *The Identification and Use of Key Characteristics in the Product Development Process*. Design Theory and Methodology Conference, ASME Design Technical Conferences, Irvine, CA.

Mei, W. H., K. E. Case, et al. (1975). "Bias and Imprecision in Variable Acceptance Sampling: Effects and Compensation." *International Journal of Production Research* 13: 327-340.

Menzefricke, E. (1984). "The Effects of Variability in Inspection Error." *Journal of Quality Technology* 16: 131-135.

Phadke, M. S. (1989). *Quality Engineering Using Robust Design*. Englewood Cliffs, New Jersey, PTR Prentice-Hall Inc.

Rabinowitz, G. and H. Emmons (1997). "Optimal and Heuristic Inspection Schedules for Multistage Production Systems." *IIE Transactions* 29: 1063-1071.

Slocum, A. H. (1992). Design and testing of a high precision linear motion carriage for laser printers. MIT, Cambridge MA, 02139.

Srinivasan, V. (1998). On Interpreting Key Characteristics. New York, IBM Research.

Suh, N. P. (1990). *The Principles of Design*. New York, Oxford University Press.

Szykman, S. and J. Cagan (1997). "Constrained three-dimensional component layout using simulated annealing." *Journal of Mechanical Design, Transactions of the ASME* 119(1): 28-35.

Thornton, A. C. (1997). *Using Key Characteristics to Balance Cost and Quality During Product Development*. Design Theory and Methodology Conference, ASME Design Technical Conferences, Sacramento CA.

Thornton, A. C. (1998a). "A Mathematical Framework for the Key Characteristics Process." *Research in Engineering Design - Theory Applications and Concurrent Engineering*.

Thornton, A. C. (1998b). *Quantitative Selection of Variation Reduction Plans*. Design Theory and Methodology Conference, ASME Design Technical Conferences, Atlanta, Georgia.

Thornton, A. C. (1999). "Variation Risk Management Using Modeling and Simulation." *Journal of Mechanical Design, Transactions of the ASME*.

Thornton, A. C. and A. L. Johnson (1996). "CADET: A Software Support Tool for Constraint Processes in Embodiment Design." *Research in Engineering Design - Theory Applications and Concurrent Engineering* 8(1): 1-13.

Viswanadham, N., S. M. Sharma, et al. (1996). "Inspection Allocation in Manufacturing Systems Using Stochastic Search Techniques." *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans* 26(2): 222-230.

APPENDIX A - *KCTOOL* SOFTWARE MODULES

The purpose of this Appendix is to explain the different subroutines and modules of the *KCTool* inspection simulation. The inspection simulation utilizes the Object-Oriented programming techniques. Each product node class and KC node class were given the algorithms to perform tasks of its own and also capable of instructing other nodes to perform certain tasks. I will now describe the various tasks performed by the different subroutines.

When an inspection simulation starts, an "Run" routine is performed. It does N iteration of the product generation process. It requests a top-level product be built, and once it is built all the contained KC values be checked with an inspection routine.

```
InspectionSimulation::Run
For I = 1 to N
        For All Product Nodes
                If (NParent=0) GetProduct
        For All KC Nodes
                If (Nparent=0) Inspect
                SetOne
ShowDataToUser
```

The Run routine first find the product node at the top of the flow down by finding the product that has no parents from the list of products, and request a product be built. This request calls for the Product Node to ask for a value for each KC node that's contained within this product node, as shown below.

```
ProductNode::GetProduct
For All Contained KC Nodes
        GetOne
```

When received the request for a value, the KC Nodes will first check if this KC has already been generated. If the value has already been generated, as indicated by the bBuiltFlag being true, the KC Node will just return the value it stores. If the value has not been generated, the KC Node will send a message to the Product Node that it belongs to, and request a product be produced. The bBuiltFlag

can also be turned to false by the inspection routine; in this case, the GetOne routine will generate a new value for this KC Node.

```
KCNode:GetOne
If bBuiltFlag=false
        While pass=false
                pProductNode->MakeProduct
                pass=pProductNode->InspectProduct
return value
```

When a ProductNode make a product in MakeProduct, it calls for all the KCNodes that this Product Node contains to make (or generate) a value. This simulates the effect that when a product is made, whether by manufacturing or assembly, all the values of its KC's are defined at that point.

```
ProductNode::MakeProduct
For all contained KC Nodes
        MakeOne
```

If the KC Node has no child KC contributing to it, this means that this KC is a feature-level KC that gets generated from a manufacturing process. In this case, the KC Node MakeOne routine calls for a random number to be generated according to the normal distribution specified by the KC Node data sheet. If in the case of the KC Node being a node that has child KC's and receives contribution from them as described by the KC Connection equation (1), it requests the contributions from all its child connections.

```
KCNode::MakeOne
If (NChildren=0)
        GetRandom
Else For all contained KCConnections
        Total+=GetContribution
bBuiltFlag=true;
```

The GetContribution routine returns the value from the child KC Node multiplied by the contribution coefficient.

KCConnection::GetContribution

Contribution = C * childKCNode-

When a product is inspected, the inspection of KCs are done all at once. If any of the KC signal a "fail" flag, the whole product is either scrapped or reworked depend on the inspection setting.

ProductNode::InspectProduct

```
For All Contained KC Node
        InspectPart
        If (fail)
                If (Scrap=true) ScrapFlag=true
                If (Rework1=true) ReworkPart
                If (Rework2=true) ReworkPartToPercent
If (ScrapFlag=true) ScrapProduct
```

When a product is scrapped, all the KC nodes contained in this product node are scrapped. The KC Node scrape the part by switching the bBuiltFlag to "false" and this will cause a new value to be generated as requested by the KCNode::GetOne routine. In addition, scrapping a product requires that all the subassembly products contained within this product to be scrapped as well. This allows new values and contributions to be generated and results in a new value to be tested during next inspection. The ScrapKC routine also keeps track of how many parts are scrapped so that the simulation can keep track of the cost of scrapping.

ProductNode::ScrapProduct

```
For All Contained KC Node
        ScrapKC
ScrapProductBelow
```

ProductNode::ScrapProductBelow

```
For All Child Product Node
        ScrapProduct
```

KCNode::ScrapKC

```
bBulitFlag=false;
Nscrap+1
```

APPENDIX B - SUMMER INTERNSHIP AT COMPANY A

A summer internship was carried out at Company A[9] as part of the research. Twelve weeks were spent with one Division of the Company. Throughout the internship, I worked mainly with the supplier quality engineers and the system engineers on a new product development project.

During the first half of the internship, I worked with the *KCTool* software and tried to load their critical parameter mapping information into the software. The engineers on the project had done the identification of KCs and the flow-down of KC from the system level KCs (System CFRs) to the subsystem level KCs (Subsystem CFRs), but the flowdown from Subsystem KCs to the feature-level KCs (CTFs) has only been completed for a few subsystems. The system engineers had just gathered the information from the design engineers and collected it in a variety of spreadsheet and document formats. They were in the process of building a *Microsoft Access* database to centralize the KC flow-down information.

The SQA engineers are responsible for validating and improving the quality of the parts purchased from the suppliers. For all the parts, a one-piece-complete inspection is required to ensure all design specifications on the drawings are met, validating the supplier's ability to produce the part. In addition to the one-piece-complete, the suppliers are required to provide statistical data with at least thirty data points for all the CTFs. The purpose is to ensure the supplier's process capability is not drifting out of specification. The thirty-part-samples information is reported as Cp and Cpk values. If Cp is below 2.0 or if Cpk is below 1.5, the SQA engineers are responsible for working with the supplier, the design engineer, and the manufacturing engineer to formulate corrective actions to reduce the variation. Options of corrective actions include supplier improving the manufacturing process, supplier sorting the parts, design engineer loosening the specifications, or the Company accepting the parts as delivered with the degradation in performance of the upper level KCs.

As I tried to apply the *KCTool* application during my internship, the main problem I ran into is the lack of coefficients for the variation propagation model. Company A's mapping process only identified the existence of links between KCs but not the quantitative relation between different-level KCs. Without the quantitative coefficients to describe how the lower-level KCs propagate variations to upper-level KCs, the flow-down model is incomplete. As a result the Monte Carlo simulations and the optimization functions of the *KCTool* software cannot be applied.

---

[9] Name disguised to protect proprietary information.

Starting about halfway through the internship, I took on the task of upgrading the information system for the SQA engineers. SQA engineers were using three separate MS Excel sheets (Open Log, Closed Log, and the CTF Log) to track the quality of the CTFs on parts from suppliers. The MS Excel system meets the SQA engineers' need, but it is difficult to integrate the information with the mapping process, since *MS Excel* is not capable of handling the many-to-many relationships of the mapping information. After the discussion with the system engineers, it was agreed that the SQA database and the mapping information could be integrated into one central database. Considering the needs of both the SQA engineers and the system engineers, I constructed an integrated Critical Parameter Management Database, which resides on the Division server. Using the client/server database model, both the SQA engineers and the system engineers can access the same database simultaneously from the computers on their desks. The system engineers are responsible for constructing the flow-down information from system CFRs subsystem CFRs to CTFs, and the SQA engineers are responsible for updating the CTF quality information. Since the both the flow-down mapping information and the CTF quality information resides in the same database on the server. Everyone, including the design engineers and the manufacturing engineers, can look up the most up-to-date variation propagation contributions for one or all CFRs and also the most up-to-date process capability from the suppliers. The user security was set up giving the SQA engineers and the system engineers the permission to modify information that each is responsible for, but everyone has the permission to view the flowdown and the process capability data. At the end of the internship, a presentation was done to train all engineers on the project how to access information on the database. Separate presentations were also given to senior managers. A user manual was developed to help new users learn to use the database system.