ABSTRACT

Title of Dissertation: CONCURRENT MULTI-PART MULTI-EVENT DESIGN REFRESH PLANNING MODELS INCORPORATING SOLUTION REQUIREMENTS AND PART-UNIQUE TEMPORAL CONSTRAINTS

Raymond Stanley Nelson III, Doctor of Philosophy, 2012

Dissertation Directed By: Professor Peter A. Sandborn
Department of Mechanical Engineering

Technology obsolescence, also known as DMSMS (Diminishing Manufacturing Sources and Material Shortages), is a significant problem for systems whose operational life is much longer than the procurement lifetimes of their constitute components. The most severely affected systems are sustainment-dominated, which means their long-term sustainment (life-cycle) costs significantly exceed the procurement cost for the system. Unlike high-volume commercial products, these sustainment-dominated systems may require design refreshes to simply remain manufacturable and supportable. A strategic method for reducing the life-cycle cost impact of DMSMS is called refresh planning. The goal of refresh planning is to determine when design refreshes should occur (or what the frequency of refreshes should be) and how to manage the system components that are obsolete or soon to be obsolete at the design refreshes.

Existing strategic management approaches focus on methods for determining design refresh dates. While creating a set of feasible design refresh plans is achievable using existing design refresh planning methodologies, the generated refresh plans may not satisfy the needs of the designers (sustainers and customers) because they do not conform to the constraints imposed on the system.

This dissertation develops a new refresh planning model that satisfies refresh structure requirements (i.e., requirements that constrain the form of the refresh plan to be periodic) and develops and presents the definition, generalization, synthesis and application of part-unique temporal constraints in the design refresh planning process for systems impacted by DMSMS-type obsolescence.

Periodic refresh plans are required by applications that are refresh deployment constrained such as ships and submarines (e.g., only a finite number of dry docks are available to refresh systems). The new refresh planning model developed in this dissertation requires 50% less data and runs 50% faster than the existing state-of-the-art discrete event simulation solutions for problems where a periodic refresh solution is required.

CONCURRENT MULTI-PART MULTI-EVENT DESIGN REFRESH PLANNING MODELS INCORPORATING SOLUTION REQUIREMENTS AND PART-UNIQUE TEMPORAL CONSTRAINTS

by

Raymond Stanley Nelson III

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2012

Advisory Committee:
Professor Peter A. Sandborn, Chair
Professor Donald Barker
Professor Shapour Azarm
Associate Professor Jeffrey W. Herrmann
Professor Bilal M. Ayyub

DEDICATION

I dedicate this dissertation to my son Roman.

ACKNOWLEDGEMENTS

I'd like to thank my advisor Dr. Peter Sandborn whose guidance, wisdom and dedication to my academic career these past many years has been a pivotal drive to attaining this great achievement. I will be forever grateful.

I would also like to acknowledge the National Science Foundation (Division of Design and Manufacturing Innovation) Grant Nos. CMMI 928628, 928837 and 928530 for their support. Finally, I would also like to thank the more than 100 companies and organizations that support research activities at the Center for Advanced Life Cycle Engineering at the University of Maryland annually.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLTURE

a_k = The lower bound of constraint period *k*. This would mean there are *K* total constraints. *k* would represent the constraint period index.

b_k = The upper bound of constraint period *k*. This would mean there are *K* total constraints. *k* would represent the constraint period index.

C_{BB_i} = The total bridge buy cost for component *i*

c_{board} = Cost of Refreshing Board: is the cost of performing a design refresh on a board (i.e., also called assembly)

C_{DR_0} = The design refresh cost in year zero

C_m = Recurring cost of manufacturing a system instance at the *m*th manufacturing event, including spares

$c_{maintenance}$ = Cost of Maintenance: the recurring cost of activities and materials needed to facilitate the operation of the obsolescence cost mitigation solution

$c_{operation}$ = Cost of Operation: the recurring cost of utilities needed to facilitate the operation of the obsolescence cost mitigation solution.

c_{part} = Cost of Refreshing Part: is the cost of performing a design refresh on a part (i.e., also called component)

$c_{production}$ = Cost of Producing Sub-Term: is the cost of manufacturing new instances of system managed

c_{spares} = Cost of Spares Sub-Term: is the cost of all spares of components as part of the obsolescence cost mitigation

$d_n;\ d_m$ = Difference in years between *n*th/*m*th manufacturing/design refresh event date and the base year for money

D_{pc} = The date of obsolescence for the predecessor component.

EOS = The end of support.

$f(\bar{x}, \bar{p})$ = Generalized cost function used to illustrate the point that the DRP problem can be formulated into a formal optimization form

$g_k(\bar{x}, \bar{p});\ g_k(Y_R, N)$ = Temporal (inequality) constraint k

$h(Y_R, N)$ = Equality constraint

i = Design refresh index.

χ = Constraint checking function

I = Number of parameters within the parameter vector.

I_I = The life-cycle code indicating an emerging component

I_O = The life-cycle code indicating the component is obsolete

I_R = The life-cycle code of synthesized part

j = Design refresh event date index.

k = Index used to identify a particular constraint

K = Total number of constraints

L = The procurement life of a component

LAT = The look-ahead time

m = Manufacturing event date index.

M = Number of manufacturing events

n = The design refresh period index.

N = Number of design refreshes within a design refresh plan.

N_{mc} = The number of Monte Carlo runs

N_f = The number of violated constraints

NRE_n = Non-recurring cost of the *n*th design refresh

$\bar{p}$ = Parameter vector.

P_0 = Price of obsolete component in year zero.

P_{0_i} = Price of obsolete component in year zero for component *i*

p_1 = First parameter within parameter vector.

p_i = Procurement life for component *i*

P_f = The probability of constraint violation

p_{max} = The largest procurement life out of all the unique components that make up a system.

$PSTD$ = The periodicity measure which is the standard deviation of the design refresh period times for a design refresh plan.

Q = Constant demand rate not dependent on year

Q_t = Demand rate dependent on year t

Q_m = Quantity of systems to be manufactured at the *m*th manufacturing event, including spares

r = The discount rate

t = Number of years from year zero

t_{BB} = The time when a bridge buy is made.

t_{DR} = The time when a design refresh is made.

TC_{BB} = Bridge buy cost

TC_{DR} = Design Refresh Term: represents the design refreshing approach to obsolescence cost mitigation which is one of two major methods of strategic management (the other being reactive approaches).

TC_M = Mitigation Term: represents the reactive mitigation approaches to obsolescence cost mitigation which is one of two major methods of strategic management (the other being design refreshing).

TC_O = The total cost for managing the obsolescence (total cost of ownership)

TC_S = Setup Term: represents the activities and infrastructure needed to make the design refresh possible as well as support activities such as the cost of holding large quantities of spares.

T_i = The design refresh period time value for design refresh period *i*.

μT = The mean design refresh period time for a design refresh plan

μ = The mean design refresh date

$\bar{x}$ = Design refresh plan also known here as the design variable vector.

x_i = Date of the design refresh i.

Y_R = The year of the design refresh. The wait time between component obsolescence and design refresh

Chapter: 1 Introduction

Obsolescence is defined as the loss or impending loss of original manufacturers of items or suppliers of items or raw materials [1]. The type of obsolescence addressed in this dissertation is referred to as DMSMS (Diminishing Manufacturing Sources and Material Shortages) and is caused by the unavailability of technologies or components that are needed to manufacture and/or support a product [2]. In this dissertation, "component" refers to the lowest management level possible for the system being analyzed. Electronic systems suffer the most severe obsolescence issues since electronic parts evolve quickly because their supply chains are driven by high clockspeed products [3], such as mobile phones and laptop computers. In some systems, the "components" are laptop computers, operating systems, and cables; while in other systems the components are integrated circuits (chips). DMSMS means that due to the length of the system's manufacturing and support life, coupled with unforeseen life extensions to the support of the system, needed components become unavailable (or at least unavailable from their original manufacturer) before the system's demand for them is exhausted. Component unavailability from the original manufacturer means an end of production and/or support for the component. It is possible for aftermarket suppliers to continue to sell a component after obsolescence; however not all components are available in the aftermarket and buying components in the aftermarket is expensive and introduces additional risks that may be unacceptable for many types of systems, e.g., counterfeit risk [4].

Inventory or sudden obsolescence, which is more prevalent in the operations research literature, refers to the opposite problem to DMSMS obsolescence. Inventory obsolescence occurs when the product design or system component specifications changes such that the

inventories of components are no longer required, e.g., [5]. This dissertation is working within the DMSMS obsolescence problem space and not the inventory obsolescence.

1.1 Obsolescence Management

The escalating impact of DMSMS-type obsolescence on systems has resulted in the development of a growing number of methodologies, databases and tools that address the obsolescence status of components, forecast future obsolescence risk and provide DMSMS mitigation and management support [6] [7]. Effective long-term management of DMSMS in systems requires addressing the problem on three different management levels: reactive, pro-active and strategic [8].

1.1 Sustainment-Dominated Systems

Sustainment in this dissertation refers to three things: keeping the system operational, continuing to manufacture and install versions of the original system that satisfy the original requirements, and finally the ability to manufacture and install versions of the original system that satisfy new and evolving requirements. The DMSMS-type obsolescence problem is especially prevalent in "sustainment-dominated" systems where the cost of maintaining the system over its support life far exceeds the cost of manufacturing or procuring the system [9]. Examples of sustainment-dominated systems include airplanes, power plant controls, medical systems, military systems [10], telecommunications infrastructure, and other safety- and mission- critical systems. These types of systems have long enough design cycles that a significant portion of the technology in them is obsolete prior to the system being fielded for the first time. Once in the field, their operational support can be 30 years or more [11]. For these systems, simply replacing obsolete components with newer components is often not a

viable solution because of high re-engineering costs and the prohibitive cost of system re-qualification and re-certification [12]. For example, if an electronic component in the 25-year old control system of a nuclear power plant fails, an instance of the original component may have to be used to replace it so as to not jeopardize the "grandfathered" certification of the plant.

1.1.1 Reactive DMSMS Management

Reactive management of DMSMS is concerned with determining an appropriate, immediate resolution to the problem of components that are obsolete or soon will be obsolete. Common reactive DMSMS management approaches include: lifetime buy, bridge buy, alternative or substitute parts, buying from aftermarket sources, uprating [13], emulation, and salvage [14]. For example, lifetime buy refers to buying enough components from the original manufacture prior to the component's discontinuance to support all forecasted future manufacturing and support needs, and bridge buy means buying a sufficient number of components to reach a pre-determined future date (refresh date) when the component will be designed out of the system.

1.1.2 Pro-active DMSMS Management

Pro-active management means that critical components that: a) have a risk of going obsolete, b) lack sufficient available quantity after obsolescence, and c) will be problematic to manage if/when they become obsolete; are identified and managed prior to their actual obsolescence event. Pro-active management requires an ability to forecast obsolescence risk for components [15] [16]. It also requires that there be a process for articulating, reviewing and updating the system-level DMSMS status.

1.1.3 Strategic DMSMS Management

Strategic management of DMSMS means using obsolescence data, logistics management inputs, technology forecasting, and business trending to enable strategic planning, life-cycle optimization, and business case development for the support of systems. The most common approach to DMSMS strategic management is DRP (Design Refresh Planning), which consists of choosing the best mix of design refreshes and reactive management approaches. A design refresh means replacement of one or more obsolete components with non-obsolete components in order to keep the system sustainable.[1] Between design refreshes, the system's design cannot change, i.e., manufacturing of new systems and maintenance of existing systems is allowed, but changes to the bill of materials (list of components) cannot be made.

1.2 Design Refresh Planning

The objective of DRP (Design Refresh Planning) is to determine when design refreshes should occur such that the life-cycle costs of the system are minimized. Value is usually gained from the DRP models through the identification of cost avoidance opportunities (opportunities to avoid future sustainment costs) associated with optimal planning of refreshes (optimal set of refresh dates or the optimal frequency at which to refresh a system); optimal mixing of reactive DMSMS mitigation solutions with design refreshes, or by identifying refresh points early enough that appropriate budgets and resources can be put in place.

[1] A design refresh is used as a reference to system changes that "Have To Be Done" in order for the system functionality to remain viable. A redesign is used to identify the "Want To Be Done" system changes, which include both the new technologies to accommodate system functional growth and new technologies to replace and better the existing functionality of the system [17].

Figure 1-1 identifies the inputs and outputs of the DRP process. The four main inputs to the DRP process are the BOM (Bill of Materials) of the system being managed, the forecasted obsolescence dates for the components in the bill of materials, the future demand for the system being produced and sustained (future manufacturing needs and spare parts required to maintain fielded systems), and the reactive management plan (applied between refreshes). The obsolescence forecasting that is an input to the DRP process can be performed using ordinal scale approaches [18], [19], life cycle curve forecasting [20], Logistic regression [21], data mining [22], leading indicators [23], or procurement lives [24]. Obsolescence forecasts for electronic components are also available from several commercial sources.

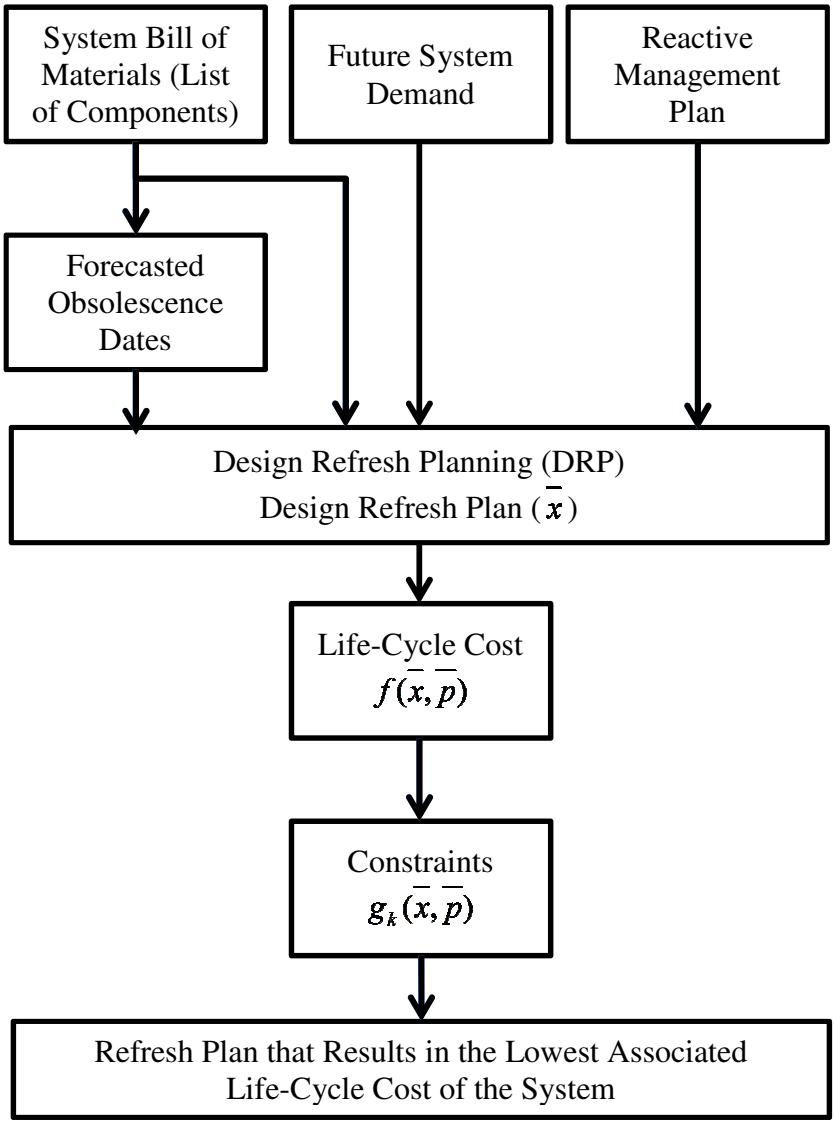


Figure 1-1: The Design Refresh Planning (DRP) process showing the required input data and the resulting output.

The BOM contains component specific information such as component quantity and cost. The BOM is also the input to an obsolescence forecasting method, the output of which are obsolescence dates for all the components in the BOM. Several approaches to DRP exist (and a new approach is developed in Chapter 2 of this dissertation). The most accurate DRP models are discrete event simulators. In discrete event simulators, the DRP process simulates a timeline of events based on the input data and generates various combinations of

design refresh dates (e.g., [28]). Each unique combination of design refresh dates is referred to as a design refresh plan. The plans are analyzed with a life-cycle cost model. The cost of a design refresh depends on the specific components it replaces and the necessary re-qualification costs – even relatively minor changes may become prohibitively expensive if system re-qualification is necessary. The design refresh plan that has the lowest associated life-cycle cost out of the selection of feasible plans is then selected.

Finding the refresh plan (combination of refreshes) with the lowest associated life-cycle cost is similar to a deterministic optimization problem where the objective function is the life-cycle cost of the system and the design variable is the refresh plan. Assuming the objective function (discussed in Section 1.3) is accurate, the only additional means of ensuring whether the refresh plan found from the least expensive search process is feasible under real-world conditions is by creating constraints that represent the restrictions placed on the design (management of the system in our case). Constraints ensure that the form of the refresh plan is acceptable and the design variable (the refresh plan dates) do not take on values that are either not allowed or impossible in the real world.

1.3 Design Refresh Problem Formulation

The DRP problem can be formulated as shown in Equation 1-1:

minimize: $f(\bar{x}, \bar{p}) = TC_M + TC_{DR} + TC_S$ **Equation 1-1**

subject to: $g_k(\bar{x}, \bar{p}) \le 0; \quad k = 1,\ldots,K$

where,

$TC_M = c_{spares} + c_{production} + c_{consumption}$	
TC_M	Mitigation Term: represents the total cost of all reactive mitigation approaches to obsolescence cost mitigation which is one of two major methods of strategic management (the other being design refreshing).[2]
c_{spares}	Cost of Spares Sub-Term: is the cost of all spares of components as part of the obsolescence cost mitigation.
$c_{production}$	Cost of Producing Sub-Term: is the cost of manufacturing new instances of system managed.
$TC_{DR} = c_{part} + c_{board} + c_{requalification}$	
TC_{DR}	Design Refresh Term: represents the total cost of the design refreshing approach to obsolescence cost mitigation which is one of two major methods of strategic management (the other being reactive approaches).
c_{part}	Cost of Refreshing Part: is the cost of performing a design refresh on a part (i.e., also called component).
c_{board}	Cost of Refreshing Board: is the cost of performing a design refresh on a board (i.e., also called assembly).
$c_{re\text{-}qualification}$	Cost of Re-qualifying: is the cost of re-qualifying refreshed system.
$TC_S = c_{operation} + c_{maintenance}$	
TC_S	Setup Term: represents the total cost of all the activities and infrastructure needed to make the design refresh possible as well as support activities such as the cost of holding large quantities of spares.
$c_{operation}$	Cost of Operation: the recurring cost of utilities needed to facilitate the operation of the obsolescence cost mitigation solution.
$c_{maintenance}$	Cost of Maintenance: the recurring cost of activities and materials needed to facilitate the operation of the obsolescence cost mitigation solution.

[2] In Chapter 2, TC_M appears as TC_{BB}, where *BB* indicates "Bridge Buy", which is a specific type of mitigation approach.

The objective function, $f(\bar{x}, \bar{p})$ calculates the LCC (Life-Cycle Cost) for the system being modeled.[3] The LCC objective function is dependent on $\bar{x} = [x_1, \ldots, x_N]$, which is the design variable vector and $\bar{p} = [p_1, \ldots, p_I]$, which is the set of parameters. The design variable is a vector of zero or more design refresh dates representing one design refresh plan.[4] It is assumed that the design variable can be changed (i.e., the design variable $\bar{x}$ can be varied to create various unique alternative refresh plans)[5]. The design variable is subject to inequality constraints, $g_k(\bar{x}, \bar{p})$.

It is assumed that the parameters ($\bar{p}$) cannot be changed. The parameters used in the LCC objective function have uncertainty; however, everything is known about the behavior and range of variation for each parameter. The quantities that are used as parameters can be the production schedule, forecasted obsolescence dates, and costs for different DRP activities. Since the quantities used in the design variable vector and the set of parameters represent monetary and quantitative amounts, $\bar{x}$ and $\bar{p}$ are restricted to real values.

1.3.1 Example of a Formulated DRP Model

An example of a formulated DRP model is the MOCA model [25]. Equation 1-1 is a summary of the MOCA model. Notice the first term of the MOCA model is essentially the

[3] While the objective function is not expressed in terms of the design variable or parameter vectors or their elements, they are included in the terms or sub-terms that make up the objective function.

[4] In theory there are in infinite number of design refresh plans that make up a design space that can be evaluated. The actual set of plans you have to select from, which are called candidate design refresh plans, will not include all feasible plans. This is because the plans you have to select from either: conform to a particular management "tradition", "style", or "culture," which may be mandated by the customer, or they were generated by an algorithm that conforms to a set of generation rules. Therefore, it is valuable to be able to choose the best plans from candidate design refresh plans where it is possible that none of the plans satisfy all the imposed constraints all the time.

[5] $\bar{x}$ is in general a vector of values; however, for the periodic DRP model developed in Chapter 2, the vector is a one dimensional, single variable.

mitigation term where the obsolescence cost is mitigated through reactive measures such as buying enough spares. The second term in Equation 1-1 is the design refresh cost term and accounts for all the costs used in performing and maintaining a design refresh.

minimize: $$f(\bar{x},\bar{p}) = \sum_{m=1}^{M} \frac{Q_m C_m}{\left(1+\frac{r}{100}\right)^{d_m}} + \sum_{n=1}^{N} \frac{NRE_n}{\left(1+\frac{r}{100}\right)^{d_n}}$$ **Equation 1-2**

subject to: $$g_k(\bar{x},\bar{p}) \le 0; \qquad k = 1,2,\ldots,K$$

Q_m	Quantity of systems to be manufactured at the *m*th manufacturing event, including spares
C_m	Recurring cost of manufacturing a system instance at the *m*th manufacturing event, including spares
NRE_n	Non-recurring cost of the *n*th design refresh
m	Index used to identify a manufacturing event.
M	Number of manufacturing events
n	Index used to identify a design refresh.
N	Number of design refreshes in the plan
r	After tax discount rate on money
d_m ; d_n	Difference in years between *n*th/*m*th manufacturing/design refresh event date and the base year for money
k	Index used to identify a constraint
K	Number of constraints

Alternate formulations of DRP models will be discussed in Chapter 2 of this dissertation.

1.3.2 Gaps

No existing design refresh planning models include any type of requirement or constraint on their solution. Most notably, the requirements that are absent in the existing DRP models are ones controlling the form of the solution, i.e., the structure of the resulting design refresh plan.

The constraints that are absent in the existing DRP models are ones controlling the feasible region of solutions, i.e., the placement of design refreshes along a timeline. These temporal constraints govern "when" refreshes must occur. This dissertation is focused on these requirements and constraints.

When considering the solution form of a design refresh plan there are two general structures: non-periodic refreshes and periodic refreshes (i.e., periodic refreshing). The requirement that a design refresh planning solution take a specific form such as periodic refreshing is common and is commonly used in DoD management. In periodic refreshing, as its name implies, the design refreshes within a design refresh plan occur at constant intervals through the support life of the system. Having a solution requirement that forces the use of periodic refresh plans reduces the complexity of the solution from a set of unique points (the number of these points could be as large as 10) on a timeline to a starting point and the design refresh period (i.e., the frequency that refreshing occurs). While refreshes can be done anytime, for many types of systems, planning, budgeting, and/or resource availability (governing deployment) requires that the form of the refresh solution has to be a periodic refresh plan.

Design refresh planning discrete event simulators, as they exists today, are able to simulate a timeline of events that include: component obsolescence and system demand

events; generate candidate design refresh plans and simulate each plan individually on the timeline; estimate the associated life-cycle costs for each candidate design refresh plan; model input uncertainty; and apply budgetary constraints. Existing design refresh planning is not able to generate temporal constraints based the effect of a component's obsolescence event has on the life-cycle management of the system; it cannot generate the same applicable constraint for synthesized replacement components throughout the life cycle of the system; it cannot calculate the probability of a design refresh plan satisfying all imposed constraints; it does not apply a penalty cost when a design refresh plan violates a constraint; and it does not have a way of selecting the best design refresh plan out of a set of design refresh plans that have all failed to satisfy all constraints.

1.4 Dissertation Objective and Research Tasks

This dissertation focuses on design refresh planning as a strategic management approach used to mitigate the impact of obsolescence on sustainment-dominated systems. Its objective is to address the formation of design refresh plans that predict the optimum frequency at which to refresh a system and to formulate a methodology for generating constraints to be imposed on the refresh planning solution in the presence of input uncertainty so that a best design refresh plan can be determined from a set of candidate design refresh plans. The research tasks associated with this dissertation are:

Task 1: Develop a DRP model based on the solution requirement that the design refreshes occur periodically. The model developed in this task must be able to account for multiple unique components with unique component price, demand, and procurement life. The model should use less input data than general (non-periodic refresh plan) DRP models that can handle multiple components.

Task 2: Evaluate the periodic design refresh plan based DRP model developed in Task 1 against a proven discrete event simulation based DRP model called MOCA (Mitigation of Obsolescence Cost Analysis) [25]. This task will evaluate the newly developed model by comparing it to the MOCA model on these attributes: accuracy (using MOCA's life-cycle costs as the accepted value), performance, and input data requirements.

Task 3: Develop general definitions of timeline constraining events that are applicable to the management of obsolescence in real systems. Create a temporal constraint taxonomy. The temporal constraint taxonomy developed in this task will be limited to the management of technology obsolescence in systems.

Task 4: Develop methods for synthesizing constraints from the timeline constraining event definitions developed in Task 3. Develop and implement a constraint evaluation process for analyses with input uncertainty. This task includes a formal analytical verification to validate the results produced from uncertainty analyses. Develop a method for determining the cost of refresh plans that violate constraints.

Task 5: Perform case studies to test, demonstrate and evaluate the new DRP model and constraints.

Chapter: 2 A Periodic Design Refresh Planning Model

This chapter presents a new refresh planning model that is designed to generate the optimum refresh frequency for a system consisting of multiple unique parts. The new model requires 50% less data and runs 50% faster than the existing state-of-the-art discrete event simulation solution for problems where a periodic refresh solution is required.

2.1 Existing DRP Models

A tradeoff made when developing any model is model detail (and accuracy) vs. reduced dependency on input data. In the case of design refresh planning models, generally models that are highly analytical require relatively little input data; however, the assumptions made in order to reduce the data requirements make for a model that captures less detail and thus may not capture the effects of the high dimensionality of the real problem.

Figure 2-1 qualitatively compares five representative design refresh planning cost models qualitatively: Porter [26], Kumar and Saranga [27], Zheng et. al. [28], Herald and Ramirez-Marquez [29] and Singh and Sandborn [25]. The horizontal-axis represents the amount and type of data the design refresh planning cost model requires. The horizontal-axis attempts to convey the relative data requirement differences between the models by listing key unique parameters from left to right, which is cumulative meaning all parameters between the beginning of the horizontal-axis and plotted point are used in the indicated cost model. The horizontal-axis also shows the variety of information used to describe the input data required for a model in two ways: parameter size (data array dimension), and the level of difficulty in obtaining the data. The vertical-axis represents the life-cycle cost model's level of detail or in other words the cost model's complexity. The vertical-axis measures model complexity by indicating the terms in each cost model. Each term represents a

contributing effect to the total life-cycle cost that is assumed to be independent from all other terms. There are three terms and each term is made up of sub-terms. Each term is assigned an upper case letter and its sub-terms are assigned the same letter but in lower case. Some cost models utilize some of the sub-terms and not all sub-terms that make up the entire term as defined in this dissertation. The vertical-axis is cumulative meaning all terms and sub-terms listed from the beginning of the vertical-axis to the plotted point are contained in the model.

Figure 2-1 is also a Venn diagram that shows how the solutions produced from each model are related to other solution sets. For example, the largest set titled "Multi-Part Multi-Event (All Refresh Plans Possible)" represents every possible refresh plan that can be created. Within the set of all possible design refresh plans is another sub-set titled "Multi-Part Multi-Event (Non-periodic Refresh Plans)", which represents a fraction of all possible all design refresh plans and may have been created using a simplifying assumption or economic heuristic method such as a just-in-time design refresh planning approach [25] [30]. To further emphasize the point, the reason why no current model lies outside the "Multi-Part Multi-Event (Non-periodic Refresh Plans)" set is because no current model searches all possible design refresh plans due to computational limitations so in an effort to reduce computational load the models utilize a design space limiting agent such as a heuristic or constraint. Within that sub-set is a sub-set titled "Single-Part Single-Event (Single Refresh Plan)" that represents all the design refresh plans that have only one design refresh per refresh plan and they are created considering only one part (e.g., electronic part such as a chip). The boundaries between sub-sets have been drawn such that the indicated cost models belong to whichever sub-set they are contained. Of course, as in all Venn diagrams sub-sets do not inherit properties from the set they are within; however, sets have all properties of the

sub-sets that are contained within them. Finally, since a DRP model cannot be considered a DRP model with just the "Lot Buy" or obsolescence mitigation term, there is one area labeled "Sparing models" that covers that space of the DRP landscape.

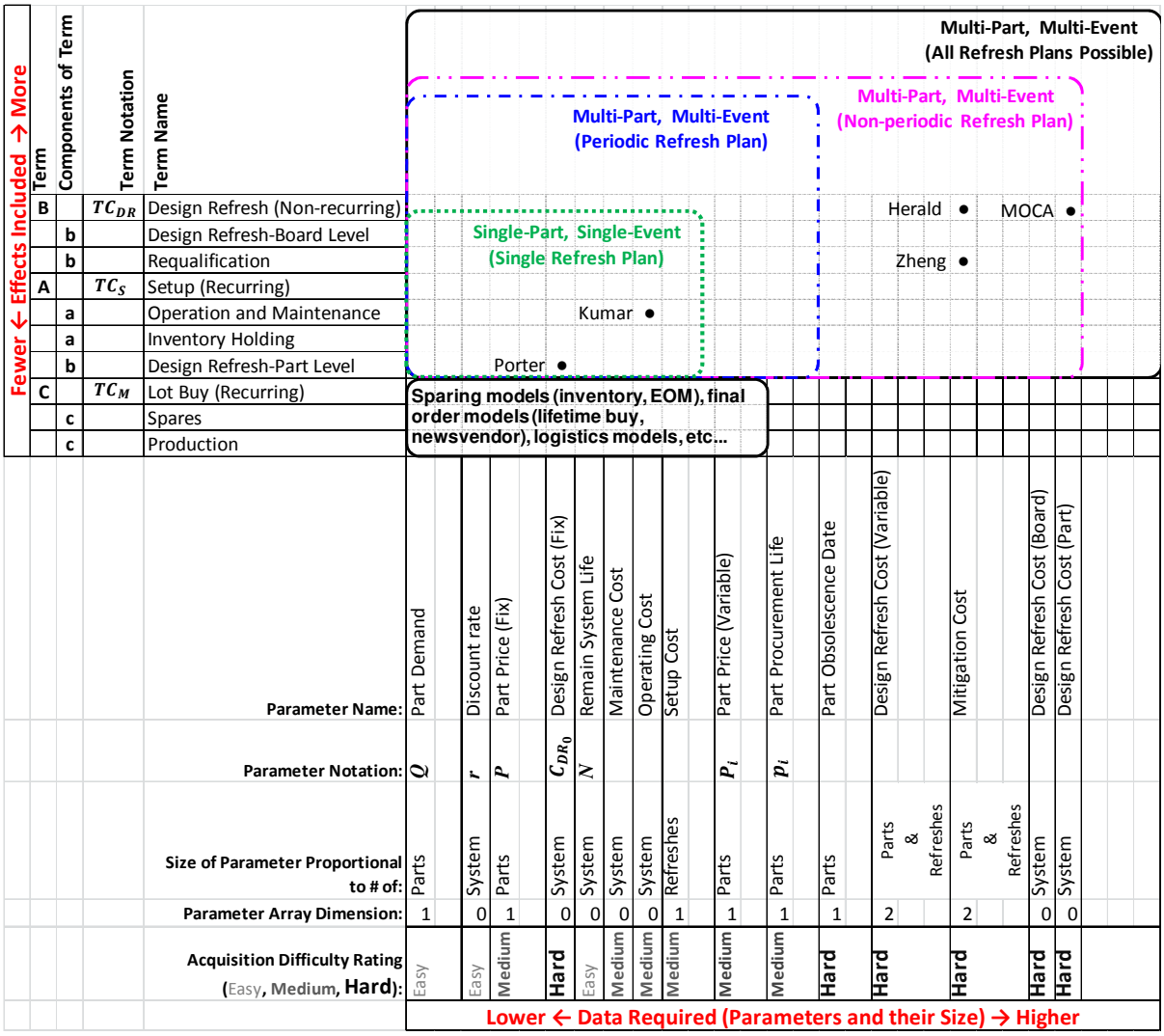


Figure 2-1: Qualitative comparison between various design refresh planning models

Some authors (e.g., Zheng) have included the Nair and Hopp model as a design refresh planning model but this is really a tech insertion[6] model and not design refresh model [31].

2.1.1 A New Class of DRP Models

In Figure 2-1 there are two groupings of models that contain existing work. The first group is made up of the Porter and Kumar models. The second group is made up of the Zheng, Herald and Ramirez-Marquez and Singh and Sandborn models. The first group of models requires a small amount of data compared with the second group; however, in order to accomplish this, these models are not very complex and can only handle one part and one design refresh event. The second group has an advantage over the first group since it can handle multiple parts and multiple design refresh events and these models are more complex with more terms; however, in order to do this they require much more data than the first group of models. There is a gap between the first and second cost model groups, where a new class of models can exist. This new class would not need much more than the data used by the first model group, and yet be able to handle multiple parts and multiple design refresh events which is a capability held only by the second model group.

Referring back to the Venn diagram, this new class of models, which is "Multi-Part Multi-Event (Periodic Refresh Plan)" is a sub-set of the "Multi-Part Multi-Event (Non-periodic Refresh Plan)"; however, it would have the "Single-Part Single-Event (Single Refresh Plan)" set as its sub-set.

[6] Technology Insertion: used to intentionally advance the capabilities of the system [17].

2.2 Moving Single-Part Single-Event to Multi-Part Multi-Event Models

The Porter [26] and Kumar and Saranga [27] models are design refresh planning cost models that consider a single refresh on a single component system and are representative of the single-part single-event model set. Porter is the more basic of the two single-part single-event design refresh cost models and it identifies two contributions to the life-cycle cost: a bridge buy and a component design refresh (See Section 1.1.1). The analysis begins at year zero with the component just as it is going obsolete, so only at this instance can a last time purchase be made for the original component. The Porter model assumes a constant yearly demand and a constant component price. The bridge buy purchases components so that the demand can be fulfilled up until the future component design refresh occurs, at some time point x. The cost of the future component design refresh is constant as well as the cost of the design refresh. All costs are added together by adjusting their value to year zero values using the net present value (NPV) function. A minimum life-cycle cost is found because the bridge buy and component design refresh contributions to the life-cycle cost of the model are increasing and decreasing respectively as x is increasing. The bridge buy contribution linearly increases as the component design refresh date (x) is pushed farther into the future. The cost-value of the component design refresh decreases as x increases since the discount rate is positive (i.e., "money today is worth less than money tomorrow"). This balance of the reactive mitigation cost (e.g., bridge buy of component) and the design refresh of the component cost is an important distinguishing characteristic of DRP models. There are other cost models that operate on different principles which are similar to DRP models in that they are balancing two diametric cost drivers (e.g., Newsvendor problem balances end-of-life buy

underage and overage costs [32], Economic Ordering Quantity problem balances ordering cost and holding/inventory cost [33]) but should not be considered a form of DRP model.

The Porter, and Kumar and Saranga models can only plan one design refresh for a system that is made up of one component. This section discusses reformulating on these models to incorporate multiple refreshes, multiple components, and with components that have different procurement lives as summarized in Table 2-1.

Table 2-1: Evolution of the Single-Part Single-Event Design Refresh Models

	Single-Part Single-Event Section 2.2.1	Single-Part Multi-Event Section 2.2.2	Multi-Event Multi-Part Section 2.2.3
Single Design Refresh	●	●	●
Single Component	●	●	●
Multiple Design Refreshes		●	●
Multiple Components			●
Multiple Component Prices			●
Multiple Procurement Lives			●

2.2.1 Multi-Part Multi-Event Model Diagram

This section will explain the MpMe periodic DRP model before it is formally formulated. Figure 2-2 is graph of inventory versus time, which is labeled to point out the mechanics of the MpMe model, making it a diagram. The origin of the graph is the beginning of the support life of the system being managed which is called year zero. It is assumed that at year zero all components are obsolete and it is also assumed that if a component is obsolete then enough inventories must be purchased until the obsolete part is refreshed out of the system and a new component is designed in so that demand can be fulfilled by buying demand as needed. So as soon as the model has begun a bridge buy is made that will fulfill the demand until the design refresh is made. This waiting time from when the components are obsolete to when a design refresh is performed is labeled as Y_R. Next, after Y_R time a design refresh is performed which is labeled as TC_{DR}. This design refresh resolves all the

obsolete components and replaces them with non-obsolete components, each of which can have unique procurement lives that do not change over time labeled as p_1, p_2 and p_3. It will be assumed that the waiting time for a design refresh will not begin until all components have gone obsolete, but before the maximum procurement life (p_{max}) expires, the shorter procurement lives run out which means when they do those obsolete components need to be bridge bought (C_{BB}) with enough inventory to last to the next design refresh. Finally, after *N* periods of Y_R and p_{max} the end of support (*EOS*) is reached; however, the components with the shorter procurement lives need the last bridge buy to last to the end of support.

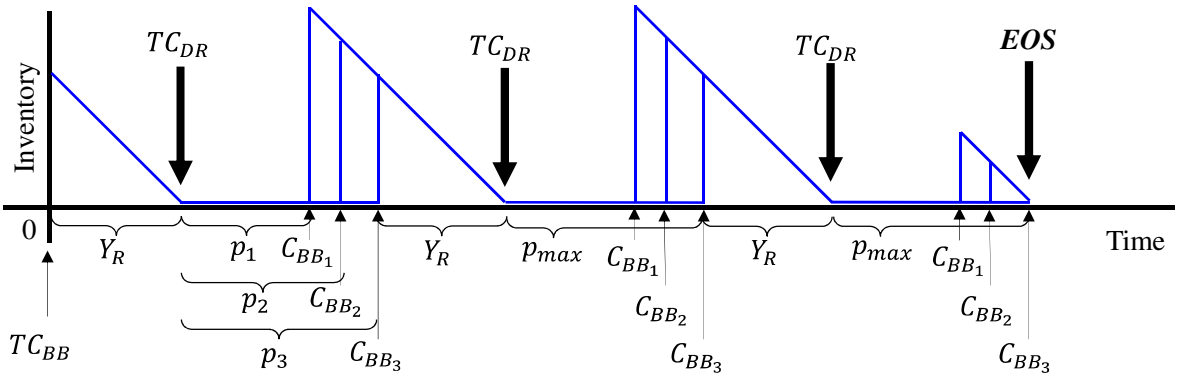


Figure 2-2: Diagram of the Multi-Part Multi-Event model

2.2.2 Single-Part Single-Event (SpSe) Model

Before presenting new DRP models the single-part single-event (SpSe) model, a thorough explanation of the single-part single-event model construction is needed. The underlying assumptions that makeup the single-part single-event model set are:

1) The system consists of a single, unique component.
2) The component will be obsolete at the beginning of year zero.
3) The annual demand for the component is constant.

4) The component and design refresh have a constant cost that is not a function of time.

The first contribution in the single-part single-event model is the cost of the bridge buy, which is the cost to purchase the number of components required to sustain or "bridge" the system until the design refresh takes place. Equation 2-1 is that general form of this term.

$$TC_{BB} = \begin{cases} 0 \text{ when } t = 0 \text{ or if } Y_R = 0 \\ P_0 \sum_{t=1}^{Y_R} Q_t \; if \; Y_R > 0 \end{cases} \quad \textbf{Equation 2-1}$$

where,

t = time in years

P_0 = the price of the obsolete component in the year of the bridge buy (year zero)

Y_R = the year of the design refresh (e.g., 0 = year of the bridge buy)

Q_t = the number of components needed in year *t*

Equation 2-1 is simplified by assuming Q_i is a constant Q (not year i dependent). Under this assumption Equation 2-1 reduces to:

$$C_{BB} = P_0 Q Y_R \quad \textbf{Equation 2-2}$$

Since the bridge buy is made in year zero and $P_0$is in year zero dollars, there is no additional computation needed.

The second contribution to the life-cycle cost calculation is the cost of the design refresh performed at time Y_R. This is simply a fixed cost (C_{DR_0}); however, since the design refresh can be performed some time after year zero, the cost of the design refresh needs to be converted back into the present value which is in the 0th year.

$$TC_{DR} = \frac{C_{DR_0}}{(1+r)^{Y_R}} \approx C_{DR_0} e^{-rY_R} \qquad \textbf{Equation 2-3}$$

where,

C_{DR_0} = the design refresh cost in year 0

r = the discount rate

The left side of Equation 2-3 assumes the interest is compounded discretely and the right side assumes interest is compounded continuously [34]; these are approximately equal when the compounding period is small (small relative to the support life of the system). The total cost for managing the obsolescence with a year Y_R refresh is given by:[7]

$$TC_O = TC_{BB} + TC_{DR} = P_0 Q Y_R + C_{DR_0} e^{-rY_R} \qquad \textbf{Equation 2-4}$$

Equation 2-4 assumes a constant annual demand and is the formulation of the Porter model [26]. In this form a closed-form analytical optimum solution can be found. Taking the derivative of Equation 2-4 with respect to Y_R, and setting it to zero, and solving for Y_R will be the optimum point in time to perform a design refresh. Equation 2-5 is the solution to the above mentioned problem and is the optimum for Equation 2-4, [35].

$$Y_R = \frac{1}{-r} \ln\left(\frac{P_0 Q}{r C_{DR_0}}\right) \qquad \textbf{Equation 2-5}$$

Although not identified as such by Porter [26], at its simplest level, the conceptual basis for the construction of the basic Porter model is similar to the construction of EOQ

[7] In Equation 1-1, TC_{BB} is replaced by TC_M. Bridge buys (*BB*) are a particular case of more general mitigation approaches.

(Economic Order Quantity) models, [36] [33]. In the case of EOQ models, the sum of the part cost (purchase price and holding/carrying cost) and the order cost is minimized to determine the optimum quantity per order. The Porter model has a similar construction where the part cost is the same as in the EOQ model (with the addition of the discount rate) and the order cost is replaced by the cost of refreshing the system to remove the obsolete part. In both cases the quantity of parts purchased is a function of the time that the stock of parts has to satisfy the demand – time waiting for a reorder (in the EOQ case) or waiting for a design refresh (in the Porter case). In the EOQ model one is generally interested in solving for the number of parts to reorder (economic order quantity) and in the Porter case one solves for the time until the refresh (economic order interval). The basic Porter model formulation uses part costs and refresh costs that are functions of time based on non-zero discount rates (which implies long time horizons), whereas basic EOQ models usually only incorporate a holding cost.

2.2.3 Single-Part Multi-Event (SpMe) Model

The SpSe model set can be extended to include multiple design refreshes by incorporating the idea of a periodic design refresh schedule, which is essentially performing a set of equally spaced design refreshes until the end of support date (*EOS*). Several assumptions in addition to the ones already made for the SpSe model must be made in order for the periodic design refresh schedule to be incorporated:

1) The life cycle of the component will begin at its design refresh and end p years later.
2) The time between the obsolescence of the component to the design refresh (i.e., Y_R) will be the same for every design refresh (this is the periodic refresh plan requirement).

3) The end of support *EOS* is equal to the product of Y_R and p periods.

4) The procurement life of the single part is constant over time.

The first contribution to the management cost is the bridge buy made at the end of the procurement life of the component. Since the SpSe model starts its analysis with the component going obsolete in year zero, a bridge buy is immediately made. Let t_{BB} denote the time the bridge buy is made, then the first contribution for the first bridge buy is:

$$TC_{BB} = \frac{P_0 Q Y_R}{(1+r)^{t_{BB}}} \qquad \textbf{Equation 2-6}$$

A general form of t_{BB} is determined by writing the first 3 bridge buy terms in a series and discerning the general pattern. Since a bridge buy is made at the end of the component's procurement life, a bridge buy is made at time zero and every time after that spaced a period of time between obsolescence events of the predecessor component and replacement component. Table 2-2 gives the first three terms for the bridge buy.

Table 2-2: First three bridge buy terms for the first three component generations

Component Generation:	Design Refresh, (n):	C_{BB}
1st	0	$\frac{P_0 Q Y_R}{(1+r)^{(0)(Y_R+p)}}$
2nd	1	$\frac{P_0 Q Y_R}{(1+r)^{(1)(Y_R+p)}}$
3rd	2	$\frac{P_0 Q Y_R}{(1+r)^{(2)(Y_R+p)}}$

From Table 2-2, the expression for t_{BB} is related to component generation in the following way:

$$t_{BB} = (n-1)(Y_R + p)$$ **Equation 2-7**

The second contribution to the management cost is the component redesign cost. Let t_{DR} denote the time the design refresh is made, then the redesign cost contribution for the first design refresh is:

$$TC_{DR} = \frac{C_{DR_0}}{(1+r)^{t_{DR}}}$$ **Equation 2-8**

Finding a general expression for t_{DR} is done in the same way as t_{BB}. Table 2-3 shows the first three design refresh terms for the first three component generations.

Table 2-3: First three design refresh terms for the first three component generations

Component Generation:	Design Refresh, (n):	C_{DR}
1st	1	$\frac{C_{DR_0}}{(1+r)^{(1)Y_R+(0)p}}$
2nd	2	$\frac{C_{DR_0}}{(1+r)^{(2)Y_R+(1)p}}$
3rd	3	$\frac{C_{DR_0}}{(1+r)^{(3)Y_R+(2)p}}$

From Table 2-3, the expression for t_{DR} is related to component generation as:

$$t_{DR} = nY_R + (n-1)p$$ **Equation 2-9**

From equations 2-6 through 2-9, the total cost over all component generations is given by:

$$TC_O = TC_{BB} + TC_{DR} = \sum_{n=1}^{N} \frac{P_0 Q Y_R}{(1+r)^{(n-1)(Y_R+p)}} + \sum_{n=1}^{N} \frac{C_{DR_0}}{(1+r)^{(nY_R+(n-1)p)}}$$ **Equation 2-10**

Equation 2-10 can be reduced by recognizing the two terms in the sums are a geometric series. Expanding the geometric series terms and reducing will yield the following equation (See Appendix A for the derivation):

$$TC_O = \left(\frac{1-(1+r)^{-N(Y_R+p)}}{1-(1+r)^{-(Y_R+p)}}\right)\left(P_0 Q Y_R + C_{DR_0}(1+r)^{-Y_R}\right) \qquad \textbf{Equation 2-11}$$

Equation 2-11 is the result of incorporating the solution requirement that the design refresh plan be periodic into the Single-Part Single-Event model. Equation 2-11 was made possible by defining the refresh period as the year to design refresh, Y_R (which has now become the time distance between the obsolescence of a part to the completion of that part's design refresh) plus the procurement life of the part, *p*, and setting the end of support date, *EOS,* equal to some multiple of the refresh period.

2.2.4 Multi-Part Multi-Event (MpMe) Model

Extending the SpMe model to include multiple components with different procurement lives, demand curves, and component prices is done by making the following additional assumptions:

1) Design refreshes are made only after every component in the system is obsolete.
2) Bridge buys are made at the end of a component's procurement life, p_i.
3) The design refresh is system-wide and affects all obsolete components.
4) The procurement lives of the multiple components are constant over time.
5) The components will be obsolete at the beginning of year zero.
6) The demand rate Q_i, and part price P_i for component i is constant over time.

7) The end of support (EOS) is equal to the sum of Y_R and p_{max} periods, $EOS = N(Y_R + p_{max})$.
8) Part holding costs are assumed to be "sunk" (i.e., costs have already been paid).

Each unique component is added to the total mitigation cost equation by modifying the bridge buy cost term to include specific price, demand, and procurement life parameters. Equation 2-12 is the revised bridge buy cost term for a unique component and is composed of two parts: the first part is the modified bridge buy contribution that replaces generalized variables with component-specific variables and a second part corrects for the initial bridge buy assumption. Since some components will go obsolete earlier than others, the bridge buy quantity is calculated to include the time the component is no longer available from the OEM specific to each component; however, at the beginning of the MpMe model, all components start as obsolete and the time that those components need to be supported for is Y_R, whereas every bridge buy made after the initial bridge buy will include different support times specific to each component.

$$C_{BB_i} = P_{0_i}Q_iY_R + P_{0_i}Q_i(Y_R + p_{max} - p_i)(1+r)^{p_{max}-p_i}\left(\frac{1-(1+r)^{-(N-1)(Y_R+p_{max})}}{(1+r)^{(Y_R+p_{max})}-1}\right) + \frac{P_{0_i}Q_i(p_{max}-p_i)}{(1+r)^{(Y_R+p_{max})N+p_i}} \quad \textbf{Equation 2-12}$$

The final term in the total mitigation cost equation is the design refresh cost term. The design refresh term given in Equation 2-13 is similar to the second term in Equation 2-11. The only change made to this equation is the use of the maximum procurement life variable, which uses the longest procurement life value from of the set of procurement lives of all the

components - this is done to reflect the assumption that no design refresh can occur until all components have gone obsolete.

$$TC_{DR} = \left(\frac{1-(1+r)^{-N(Y_R+p_{max})}}{1-(1+r)^{-(Y_R+p_{max})}}\right) C_{DR_0}(1+r)^{-Y_R} \qquad \textbf{Equation 2-13}$$

Summing all component bridge buy terms and the design refresh term gives the total mitigation cost for MpMe model, shown in Equation 2-14.

$$TC_O = TC_{BB} + TC_{DR} = \left(\sum_{i=1}^{I} C_{BB_i}\right) + TC_{DR}$$

$$= P_{0_1}Q_1Y_R + P_{0_1}Q_1(Y_R + p_{max} - p_1)(1+r)^{p_{max}-p_1}\left(\frac{1-(1+r)^{-(N-1)(Y_R+p_{max})}}{(1+r)^{(Y_R+p_{max})}-1}\right)$$

$$+\frac{P_{0_1}Q_1(p_{max}-p_1)}{(1+r)^{(Y_R+p_{max})N+p_1}}$$

$$+P_{0_2}Q_2Y_R + P_{0_2}Q_2(Y_R + p_{max} - p_2)(1+r)^{p_{max}-p_2}\left(\frac{1-(1+r)^{-(N-1)(Y_R+p_{max})}}{(1+r)^{(Y_R+p_{max})}-1}\right)$$

$$+\frac{P_{0_2}Q_2(p_{max}-p_2)}{(1+r)^{(Y_R+p_{max})N+p_2}}$$

$$+P_{0_3}Q_3Y_R + P_{0_3}Q_3(Y_R + p_{max} - p_3)(1+r)^{p_{max}-p_3}\left(\frac{1-(1+r)^{-(N-1)(Y_R+p_{max})}}{(1+r)^{(Y_R+p_{max})}-1}\right) \qquad \textbf{Equation 2-14}$$

$$+\frac{P_{0_3}Q_3(p_{max}-p_3)}{(1+r)^{(Y_R+p_{max})N+p_3}}$$

$$+\cdots$$

$$+P_{0_I}Q_IY_R + P_{0_I}Q_I(Y_R + p_{max} - p_I)(1+r)^{p_{max}-p_I}\left(\frac{1-(1+r)^{-(N-1)(Y_R+p_{max})}}{(1+r)^{(Y_R+p_{max})}-1}\right)$$

$$+\frac{P_{0_I}Q_I(p_{max}-p_I)}{(1+r)^{(Y_R+p_{max})N+p_I}}$$

$$+C_{DR_0}(1+r)^{-Y_R}\left(\frac{1-(1+r)^{-N(Y_R+p_{max})}}{1-(1+r)^{-(Y_R+p_{max})}}\right)$$

where,

$p_{max} = max(p_1, p_2, p_3, \dots, p_I)$

Unlike the single part (Sp) models the MpMe model allows the determination of periodic refresh plans for real multi-part bills of materials that constitute real systems.

At the end of Section 2.2.1 it was pointed out that the Porter model (the SpSe model) is similar in construction to economic order quantity (EOQ) models. The majority of EOQ models are constructed with the idea that only a single unit (i.e., this could be a part also known as component) was being ordered. However, extensions to the EOQ model have been made to incorporate multiple units and are known as Multi-Item EOQ models [37] [38] [39] [40] [41] [42].

The Multi-Item EOQ and MpMe models both assume that the part demand is continuous at a constant rate and that all demand for all items is satisfied on time. However, there are significant differences: the Multi-Item EOQ models assume an infinite planning horizon, whereas the MpMe model has a finite planning horizon; and the Multi-Item EOQ model assumes that all activities (for all items) take place over short time durations, which fundamentally changes the construction of the solution.

2.3 Incorporating Temporal Constraints into the MpMe Model

Temporal constraints imposed on design refresh planning usually restrict design refreshes within a defined period of time (see Section 3.2) for example, let a_k and b_k be the beginning and ending constraint bounds respectively for some constraint k. Since design refreshes occur periodically in MpMe models, all periods from year zero to the end of support need to be checked to determine whether a design refresh is present that satisfies the constraint. One way to check this is by defining a unit function χ such that given a p_{max}, if

some integer value n and floating value of Y_R exist such that $a_k \leq nY_R+(n-1)p_{max} \leq b_k$ where a_k and b_k are constraint bounds for constraint period k (See Chapter 3) then $\chi = 1$, otherwise $\chi = 0$.

$$\chi_k(Y_R, n) = \begin{cases} 1, & a_k \leq nY_R + (n-1)p_{max} \leq b_k \\ 0, & \begin{matrix} nY_R + (n-1)p_{max} < a_k \\ or \\ b_k < nY_R + (n-1)p_{max} \end{matrix} \end{cases}$$

when $a_k < b_k$

$$\sum_{n=1}^{N} \left(\sum_{k=1}^{K} \chi_k(Y_R, n) \right) \geq 1$$

for $k = 1$ to K, which is the number of inequality temporal constraints.

2.4 Verification of the MpMe Model for Specific Case Studies

This section will verify the MpMe model for specific case studies with a previously validated discrete event simulation based DRP model called MOCA (Mitigation of Obsolescence Cost Analysis). In addition to verifying the MpMe model with the MOCA model, case studies will be run to estimate the MpMe model's computational performance. Finally a discussion will be presented on the differences in data required for the MpMe as it compares to a more advanced DRP model such as the MOCA model. Table 2-4 is a summary of the case studies that will be performed on the MpMe and MOCA models.

Table 2-4: Summary of case studies performed on the MpMe and MOCA models.

		Optimization Method		Design Space		Parameters														Design Refresh Plans		Special Case	Constraints	
		Exhaustive Search	*Gradient Based*	*Continuous*	*Discrete*	*Part Demand*	*Discount rate*	*Part Price (Variable)*	*Design Refresh Cost (Fix)*	*Remain System Life*	*Part Procurement Life*	*Maintenance Cost*	*Operating Cost*	*Setup Cost*	*Part Obsolescence Date*	*Design Refresh Cost (Variable)*	*Mitigation Cost*	*Design Refresh Cost (Board)*	*Design Refresh Cost (Part)*	*Periodic*	*Non-Periodic*	*Both Models Evaluate Identical Set of Refresh Plans*	*EOS*	*Temporal*
Case 1:	MpMe	•			•	•	•	•	•	•	•									•		•	•	
	MOCA					•	•	•	•	•	•									•				
Case 2:	MpMe	•			•	•	•	•	•	•	•									•			•	
	MOCA					•	•	•	•	•	•										•			
Case 3:	MpMe	•			•	•	•	•	•	•	•									•			•	
	MOCA					•	•	•	•	•	•	•	•	•	•	•	•	•	•		•			
Case 4:	MpMe		•	•		•	•	•	•	•	•									•			•	
	MOCA					•	•	•	•	•	•										•			
Case 5:	MpMe		•	•		•	•	•	•	•	•									•			•	•
	MOCA					•	•	•	•	•	•										•			

The objectives of these case studies are as follows:

- For the same cost inputs, show that the MpMe model matches the MOCA model's life-cycle cost with respect to the design refresh plan.
- Show that the design refresh plans generated by the MpMe model are members of the set of all refresh plans generated by MOCA
- Show that the optimum periodic design refresh plan generated by the MpMe model is the optimum periodic refresh plan
- Assess whether inclusion of addition cost modeling detail (in MOCA) affects the optimum periodic refresh plan
- Assess how much different (better) the free range design refresh plan is than the optimum period plan
- Measure the relative performance of the MpMe model over the MOCA model

- Qualitatively compare the difference in data required of the MpMe model with the MOCA model

In order to compare the MpMe and MOCA models fairly, both models were implemented in the same programming language called Matlab. For performance metrics like elapsed processing time and the number of function calls, this was an especially important and necessary step.

2.4.1 Case 1: Evaluating Identical Design Refresh Plans

For this case study, both the MpMe and MOCA models will be evaluating the same set of design refresh plans with the same set of global parameters. The global parameters include all unique input component data (e.g., component price) and system data (e.g., design refresh cost). This case study is evaluating the accuracy of the MpMe model using the output from the MOCA model as the true value. Since all inputs to both models are the same, we would expect to see the same life-cycle costs for the corresponding design refresh plans. Table 2-5 and Table 2-6 show the input data used for all cases except case 3.

Table 2-5: Input data for cases 1, 2, 4 and 5.

Parameter	*Part #*	Price	Demand Rate	Procurement Life
Unit		[$]	[Qty/Yr]	[Yr]
Symbol	i	P	Q_i	p_i
Value	*1*	1.00	1	5
	2	1.00	1	10

Table 2-6: Iput data for cases 1, 2, 4 and 5.

Parameter	Discount Rate	End of Support	Fixed Design Refresh Cost (System Level)
Unit		[Yr]	[$]
Symbol	r	EOS	C_f
Value	0.095	60	60

After running both the MpMe and MOCA models with identical input data and component data the resulting Figure 2-3 was produced. Since both model's data points are overlapping completely in Figure 2-3 it indicates that MpMe model is verified against the MOCA model for the given inputs.

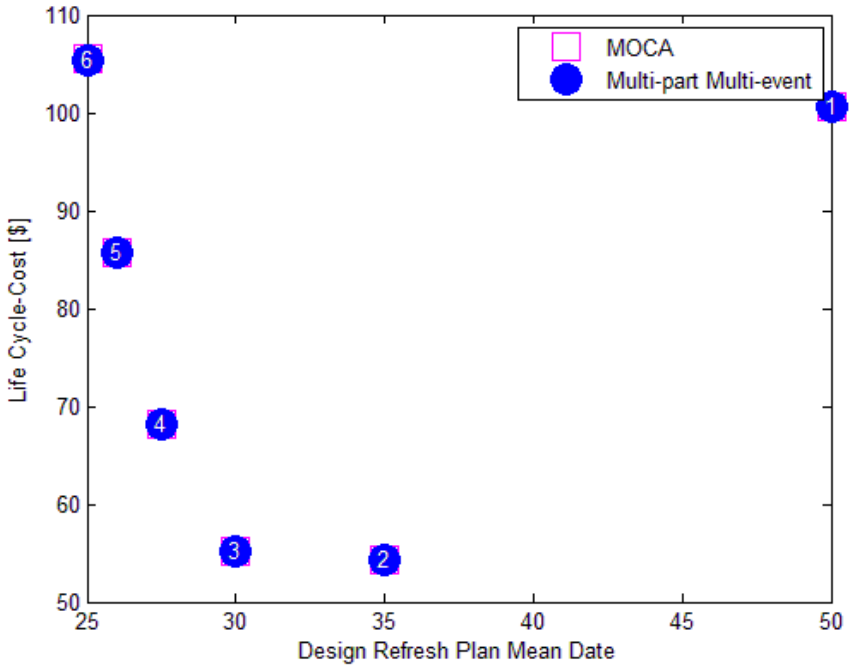


Figure 2-3: Case 1 results from MpMe and MOCA models showing all design refresh plans input into both the MpMe and MOCA models

Figure 2-4 plots the optimum design refresh plans for a given number of design refreshes per design refresh plan. In this case, since there were only 6 design refresh plans

evaluated by each model, Figure 2-4 seems redundant; however, in the following cases the type of graph shown in Figure 2-4 will prove to be more useful when the number of design refresh plans evaluated increase dramatically. It is important to note that the optimum design refresh plan for both the MpMe and MOCA models is circled using a thick, black line.

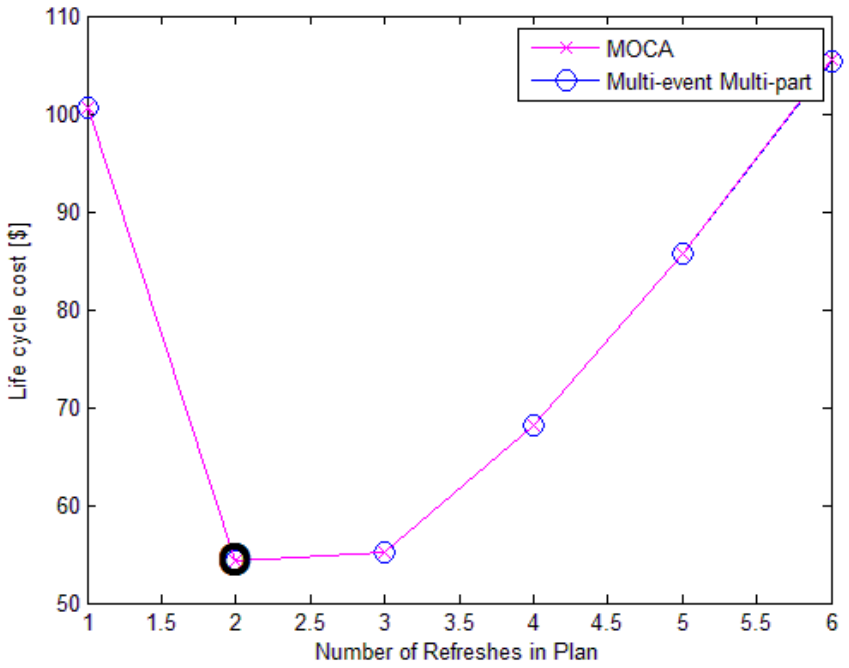


Figure 2-4: Case study 1 results for both MpMe and MOCA models showing only the design refresh plans with the lowest associated life-cycle cost for a given number of refreshes per plan.

Table 2-7 and Table 2-8 provide more detail of the optimum design refresh plans for a given number of design refreshes per design refresh plan. In this case, you can see from Table 2-7 that the lowest life-cycle cost (TC_O) was for the design refresh plan that has two design refreshes which take place on year 20 and year 50. The first thing to notice when looking at a table like Table 2-7 is how the MpMe model distributes its design refresh dates for each of its design refresh plans. Table 2-8 shows the same kind of data for the MOCA model; however, since for this case both models were forced to evaluate the same plans, both have the same optimal design refresh distribution. Every case after case 1 will show that MOCA will have a very different optimal design

refresh date distribution from the MpMe model, due to the lack of the periodic design refresh solution requirement.

Table 2-7: Case 1 results from the MpMe model showing the lowest associated life-cycle cost for a given number of design refreshes per refresh plan

	Optimal MpMe Design Refresh Plans, $\bar{x}$												TC_O[$]
N	**0**	**2**	**5**	**10**	**14**	**20**	**26**	**30**	**35**	**38**	**40**	**50**	
1												50	100.66
2						***20***						***50***	***54.32***
3				10				30				50	55.17
4			5			20			35			50	68.07
5		2			14		26			38		50	85.63
6	0			10		20		30			40	50	105.44

Table 2-8: Case 1 results from the MpMe model showing the lowest associated life-cycle cost for a given number of design refreshes per refresh plan

	Optimal MOCA Design Refresh Plans, $\bar{x}$												TC_O[$]
N	**0**	**2**	**5**	**10**	**14**	**20**	**26**	**30**	**35**	**38**	**40**	**50**	
1												50	100.68
2						***20***						***50***	***54.35***
3				10				30				50	55.19
4			5			20			35			50	68.09
5		2			14		26			38		50	85.65
6	0			10		20		30			40	50	105.46

After reviewing Table 2-7 and Table 2-8, it can be seen that the life-cycle costs of the MpMe model have an error of about $0.02, which is not significant. This error may be the result of how the computer programming languages memory allocation size per computation (e.g., round-off error) or how the interest is calculated (i.e., continuous or discrete compounding).

2.4.2 Case 1: Discussion on Performance

The MpMe model was created to reduce the input data needed to run Multi-Part Multi-Event models; however, in addition to reducing the input data the processing time has also been reduced.

The same test case conducted to verify the MpMe model with MOCA in case 1 was performed varying the end of support date from 30 to 80 years in 10 year increments. The end of support dictates how many design refreshes can take place within the support time (i.e., between year zero and *EOS*). Each time the end of support was changed, the test case was performed 1000 times, each time the processing time for each model was measured (MOCA and MpMe are analyzing the same set of design refresh plans in each case). The resulting processing times were recorded and compiled to create a graph of mean processing times for each design refresh plan evaluated. Error bars were also plotted at each data point to show 2 standard deviations from the mean of processing times, which make up approximately 95% of the processing times. The results show (Figure 2-5) in all cases that the MpMe model has processing time savings with some that are 2 times faster.

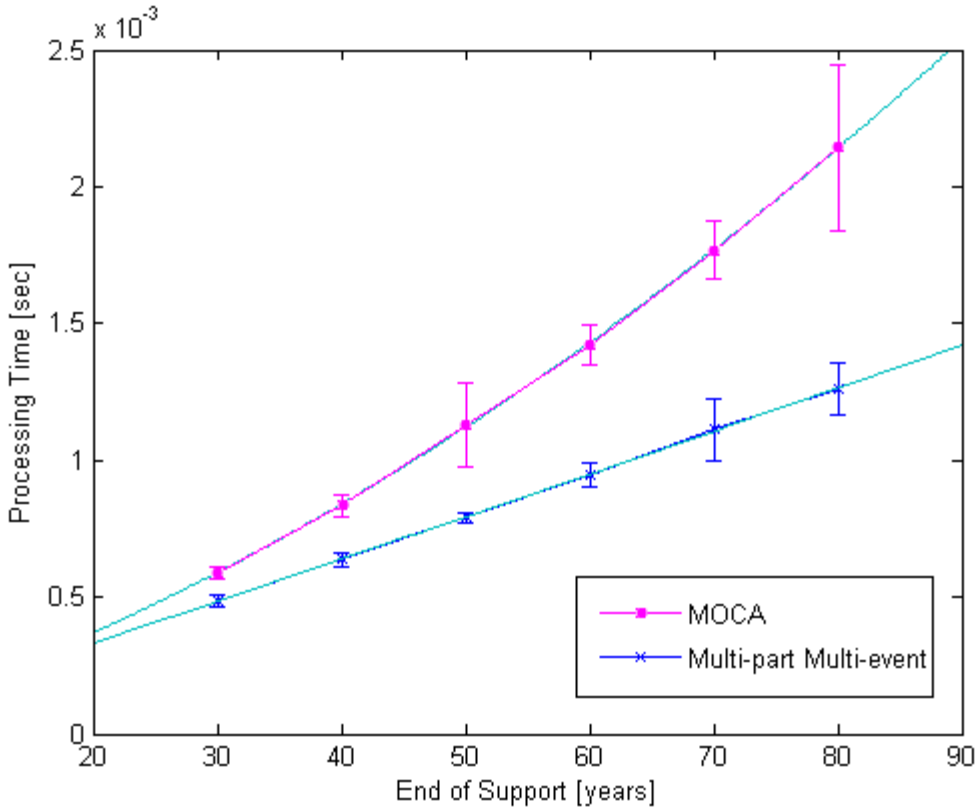


Figure 2-5: Solution processing time for MOCA and MpMe models.

Equation 2-15 and Equation 2-16 are the quadratic curve fits for solution processing time data of the MOCA and MpMe models respectively, where *EOS* is the end of support date in years.

$$T_{MOCA}(EOS) = (0.15E-6)EOS^2 + (1.45E-5)EOS + 1.98E-5 \quad \textbf{Equation 2-15}$$

$$\frac{dT_{MOCA}}{dEOS} = (0.30E-6)EOS + (1.45E-5)$$

$$T_{MpMe}(EOS) = (0.00415E-6)EOS^2 + (1.51E-5)EOS + 2.66E-5 \quad \textbf{Equation 2-16}$$

$$\frac{dT_{MpMe}}{dEOS} = (0.0083E-6)EOS + (1.51E-5)$$

Taking the derivative of both solution processing time curves allows for their empirical growth rates to be compared. Looking at both slope equations for the cost models reveals that the MOCA model is growing 36 times faster than the MpMe model, which means significant processing time savings as the end of support of the problem is extended.

2.4.3 Case 2: Combinatorial Exhaustive Search

For this case, the input data is exactly the same as case 1; however, the MOCA model is allowed to perform a combinatorial exhaustive search using the same set of unique design refresh dates that the MpMe uses. The MpMe model determines the unique set of design refresh dates by using Equation 2-17.

$$EOS = N(Y_R + p_{max}) \qquad \textbf{Equation 2-17}$$

Equation 2-17 is the mathematical relation of the 3rd assumption given in Section 2.2.2. The end of support (*EOS*) is given and the largest procurement life (p_{max}) is also known so Y_R is determined by varying *N* (Note: *N* is the number of $Y_R + p_{max}$ periods which means *N* is an integer). As long as the value of Y_R is non-negative and *N* is a non-negative integer, then there can be a range of *N* and Y_R values that will satisfy Equation 2-17.

Figure 2-6 shows the results from running the MpMe and MOCA with the same inputs given in case 1, but in this case MOCA was allowed to perform a combinatorial exhaustive search using the same unique set of design refresh dates used in the MpMe model. Neither model has an advantage over when the dates occur, just the combination of the dates in the unique set. Notice though that while MOCA is allowed to do a combinatorial

exhaustive search that it still calculates the same combination of design refreshes that the MpMe model does (e.g., In Figure 2-6, the design refresh plan with one refresh per refresh plan evaluated by the MpMe model is overlapping the same plan evaluated by MOCA during its exhaustive search).

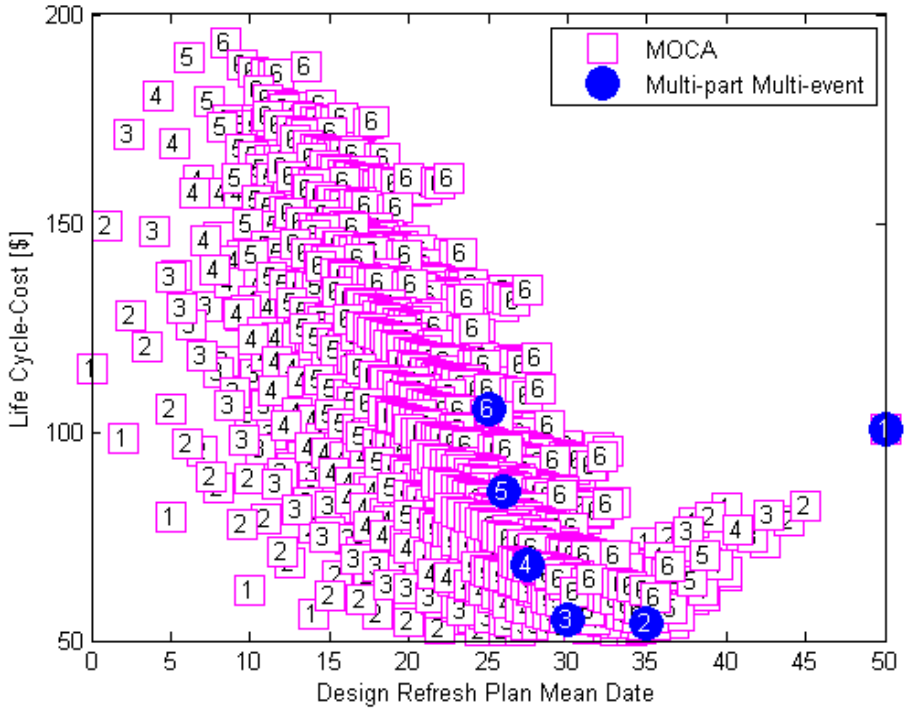


Figure 2-6: Case 2 results from MpMe and MOCA models showing all design refresh plans input into both the MpMe and MOCA models

Figure 2-7 shows the optimal design refresh plans from both the MpMe and MOCA models. Notice that for all design refresh plans, MOCA finds a cheaper refresh plan; however, despite this the MpMe model still shows the same behavior as the MOCA model in that it too finds that a design refresh plan with 2 refreshes per plan is optimal.

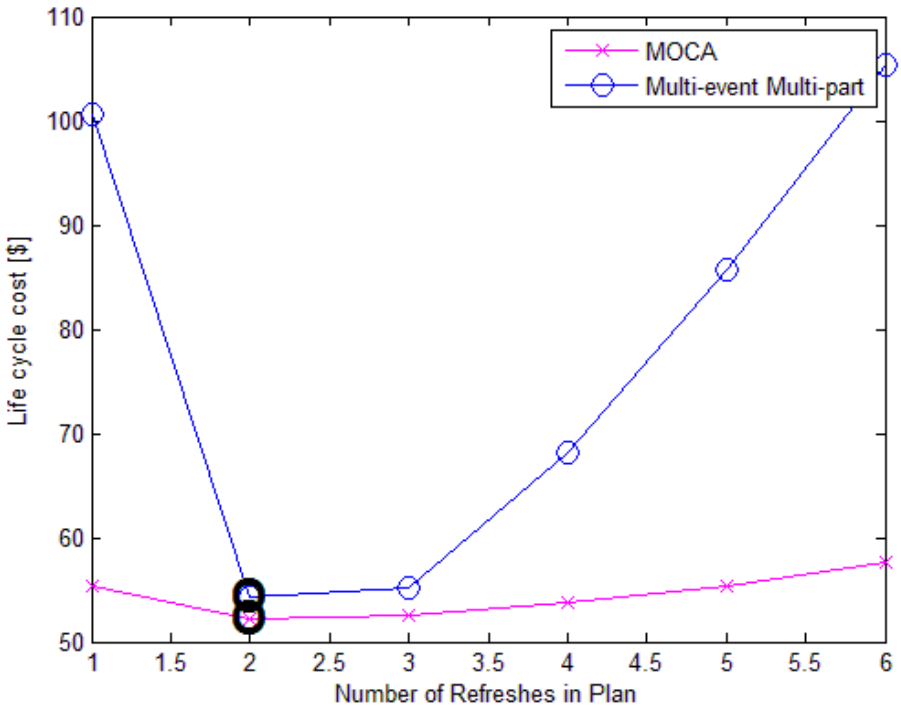


Figure 2-7: Case study 2 results for both MpMe and MOCA models showing only the design refresh plans with the lowest associated life-cycle cost for a given number of refreshes per plan.

Looking at Table 2-9 and Table 2-10 reveal that the optimal design refresh plan with 2 refreshes for both the MpMe and MOCA model are slightly similar in terms of the placement of the refreshes with respect to the entire support life; but the dates differ by at least ±6 years.

Despite this difference in the design refresh dates, the life-cycle costs of the two optimums are $2.10, which is about 3.8% difference, which is not significant.

Table 2-9: Case 2 results from the MpMe model showing the lowest associated life-cycle cost for a given number of design refreshes per refresh plan.

	Optimal MpMe Design Refresh Plans, $\bar{x}$												TC_0[$]
N	**0**	**2**	**5**	**10**	**14**	**20**	**26**	**30**	**35**	**38**	**40**	**50**	
1												50	100.66
2						***20***						***50***	***54.32***
3				10				30				50	55.17
4			5			20			35			50	68.07
5		2			14		26			38		50	85.63
6	0			10		20		30			40	50	105.44

Table 2-10: Case 2 results from the MpMe model showing the lowest associated life-cycle cost for a given number of design refreshes per refresh plan.

	Optimal MOCA Design Refresh Plans, $\bar{x}$												TC_0[$]
N	**0**	**2**	**5**	**10**	**14**	**20**	**26**	**30**	**35**	**38**	**40**	**50**	
1						20							55.36
2					***14***					***38***			***52.22***
3					14				35			50	52.47
4					14				35		40	50	53.71
5					14				35	38	40	50	55.37
6					14			30	35	38	40	50	57.72

2.4.4 Case 2: Discussion on Global Optima

Determining the existence of and finding the global optima of the unconstrained MOCA cost model are important achievements since the MOCA global optima provides a target to which we can compare the global optimum of the MpMe model. Symbolic (i.e., analytical) means of finding the MOCA global optimum are not applicable so a numerical method of finding the global optimum will be performed using an exhaustive approach. A

numerical approach to finding the global optimum may not definitively prove that the found optimum is in fact global, so in this case the combinatorial exhaustive search can be used to estimate the global optimum. It is important to keep in mind we are estimating the MOCA global optimum for the purpose of being able to compare it with the MpMe global optimum and while doing so we have to balance practicality with fairness. For practicality, the number of function calls needs to be low enough that process times do not last for an unreasonably long duration. This means we need to limit the number of locations the design refreshes can be placed from every location possible to just a set of locations. Usually, this task of limiting the available locations where design refreshes can be positioned is done by the just-in-time heuristic [25], which sets the available design refresh locations to be immediately before the beginning of a scheduled demand event such as production of a new instance of a system. The just-in-time heuristic is acceptable for the MOCA model, but not for the MpMe model since the MpMe model does not consider a scheduled demand. This is where practicality is balanced with fairness. A fair comparison of the MOCA and MpMe models is achieved by ensuring both models use the same locations for the positioning of design refreshes within a design refresh plan. The locations the MpMe model uses to position its design refresh are set based on a set of assumptions (See Section 2.2.3) and not a heuristic. So we will take the locations generated for the MpMe model that were created based on a set of assumptions and force MOCA to use only those locations when it is generating its design refresh plans to evaluate. In other words, the MpMe model will act as a means to discretize the design space in an effort to reduce the computational effort needed to estimate the MOCA global optimum while ensuring a fair comparison to the global optimum of the MpMe model.

Figure 2-6 displays the results after running the MOCA model using the validation test case parameters and giving it the same design refresh locations that the MpMe used for the same validation test case. The Figure 2-6 horizontal-axis is the mean date of all the design refresh dates that are in a design refresh plan. The vertical-axis is the life-cycle cost associated with the design refresh plan. The square data points are each a design refresh plan created by the MOCA model. The circular data points are refresh plans produced by the MpMe model. There is a number shown within the plotted data points what indicates how many refreshes are within the plan. For case 2 as it is in case 1, the end of support is 60 years. All the other parameters such as component information and discount rate are the same as case 1, 4 and 5. Notice as the average design refresh plan date increases, the overall life-cycle cost decreases and minimizes around 35 years and then begins to increase after 35 years. This suggests the MOCA model is convex and has a global optimum. Notice the MpMe model also has a global optimum around 35 years and that its global optimum is very close the MOCA global optimum which is also 35 years. The optimum MOCA plan has 2 refreshes in its plan and so does the MpMe model.

2.4.5 Case 3: Actual Example System

For cases 1 and 2, a fictional example system was used to verify and evaluate the MpMe model against the MOCA model. The emphasis for case 1 and 2 was to make the fictional example system small (i.e., 2 unique parts made up the system) so that derivatives could easily be performed by hand for the gradient based optimization schemes used in cases 4 and 5 (as well as the convexity study done in Section 2.7.7), which consequently allows for direct comparison with the combinatorial exhaustive search methods used in cases 1 and 2. For case 3, an actual example system consisting of 152 unique components will be input into

the MpMe and MOCA models and processed. Afterwards a discussion on the MpMe model accuracy and performance will be presented.

Figure 2-8 shows the distribution of all the procurement lives for the population of components used in case 3. Notice that the predominant procurement life is between 2 and 3 years and the largest procurement life is between 8 and 9 years. Looking back as Section 2.2.1 which discussed the formulation of the MpMe model diagrammatically, it be conjectured that a maximum procurement life of 9 will ultimately result in an *EOS* that is a multiple of 9. Refering to Table 2-11, the *EOS* for this case is 54 and this agrees with what we would expect.

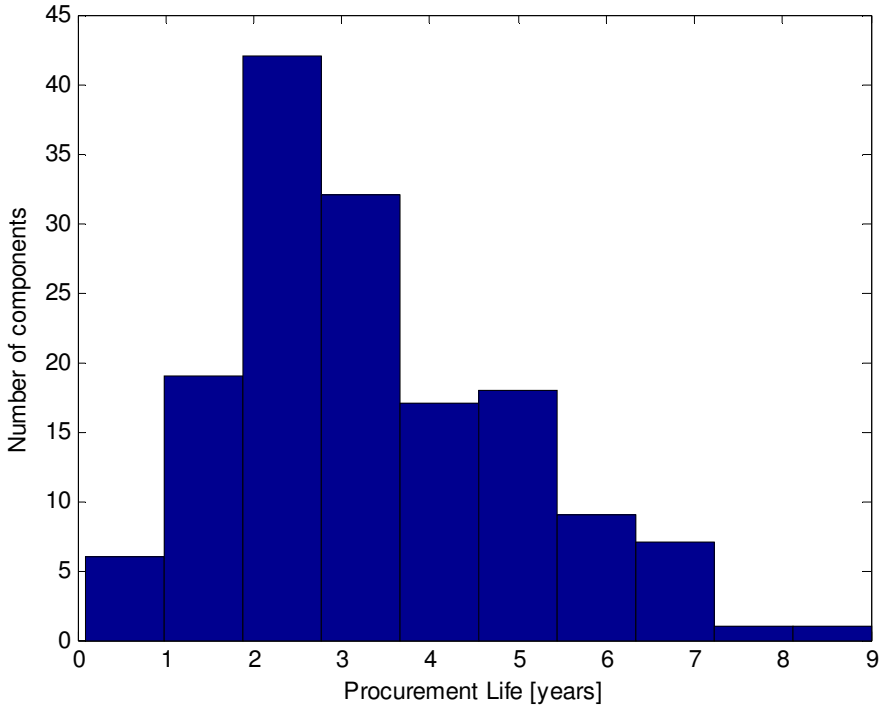


Figure 2-8: Histogram of the procurement lives of all the unique components input into the MpMe and MOCA models.

Figure 2-9 shows the distribution of component prices used in this case. Most of the component population has a price ranging from $0 to about $2500 and that the largest component price is about $26,000.

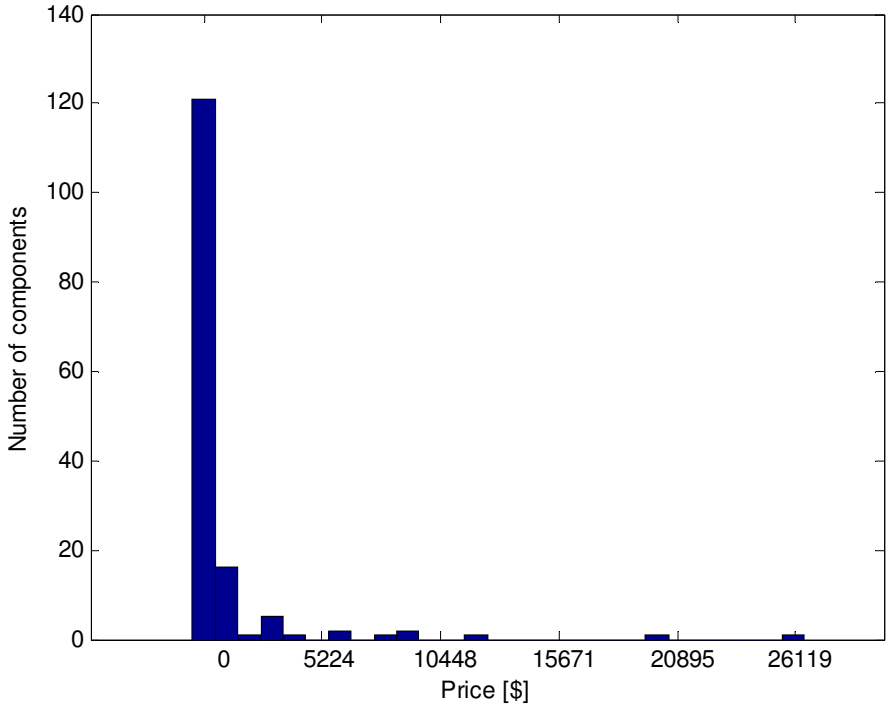


Figure 2-9: A histogram of prices for the components input into the MpMe and MOCA models.

Table 2-11 and Table 2-12 are the global parameters used in only case 3. The parameters marked with an asterisk are unique only to MOCA and are not used in the MpMe model. For case 3, an actual example system will be input into the MpMe and MOCA models; however, we would also like to know how letting MOCA use all of its parameters affect the accuracy of the MpMe model as well as the optimum MOCA solution. The first half of case 3 will be done using only the parameters that both the MpMe and MOCA share, while all the unique, MOCA parameters are omitted. The second half of case 3 will allow

MOCA to use the parameters common to both models as well as the parameters unique only to MOCA and compare the results.

Table 2-11: Input data used for case 3 (Note: Parameters are unique to MOCA are marked with a *).

Parameter	Discount Rate	End of Support	Fixed Design Refresh Cost (System Level)	Modifier on board refresh*	Design Refresh Cost (Board Level)*	Modifier on part refresh*
Unit		[Yr]	[$]		[$]	
Symbol	r	EOS	C_f	M_b	C_b	M_p
Value	0.095	54	5M	1	4M	1

Table 2-12: Input data used for case 3 (Note: Parameters are unique to MOCA are marked with a *).

Parameter	Design Refresh Cost (Part Level)*	Re-qualification Cost*	Modifier on effective part price*	Storage Handling Fraction*
Unit	[$]			
Symbol	C_p	C_Q	M_i	
Value	100k	25k	1.2	0.00

Case 3: First Half – Parameters Common Between MpMe and MOCA are Used

Figure 2-10 shows that the MpMe model is capable to handling just as many components as the MOCA model while also placing its optimum number of refreshes per refresh plan close to the MOCA optimum. In this case, the MOCA optimum number of design refreshes per refresh plan appears to be 2 and the mean design refresh date is somewhere in the range of around 18 to 33 years. The MpMe model more clearly shows an optimum of 2 design refreshes per refresh plan and an optimum mean refresh date of 32 years.

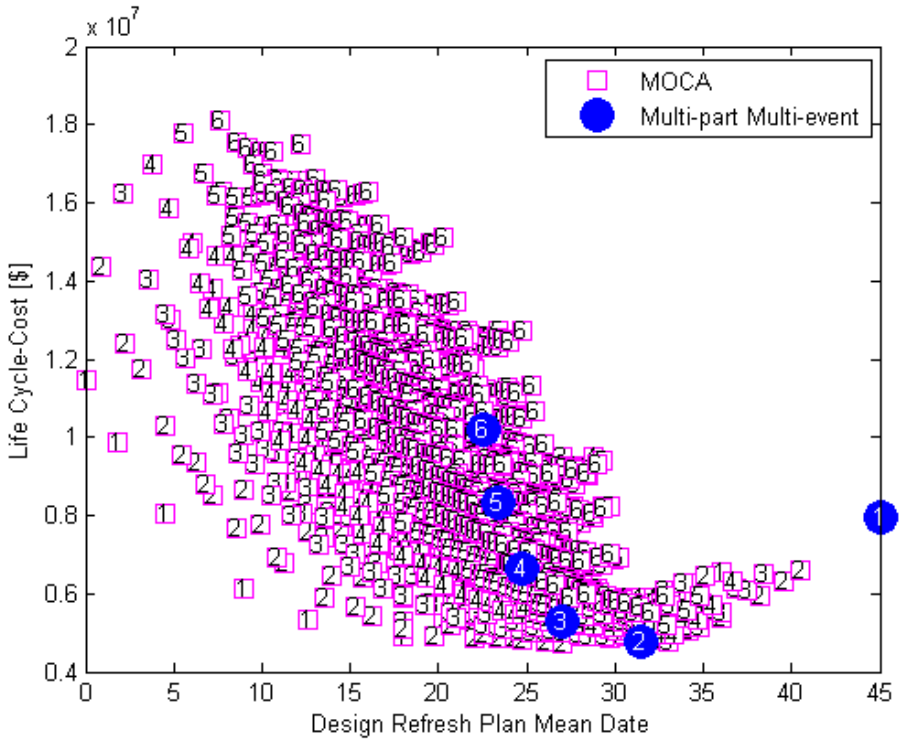


Figure 2-10: Case 3 results from MpMe and MOCA models using common parameters

Figure 2-11 confirms the observation that both models have an the optimum number of design refreshes per refresh plan at 2.

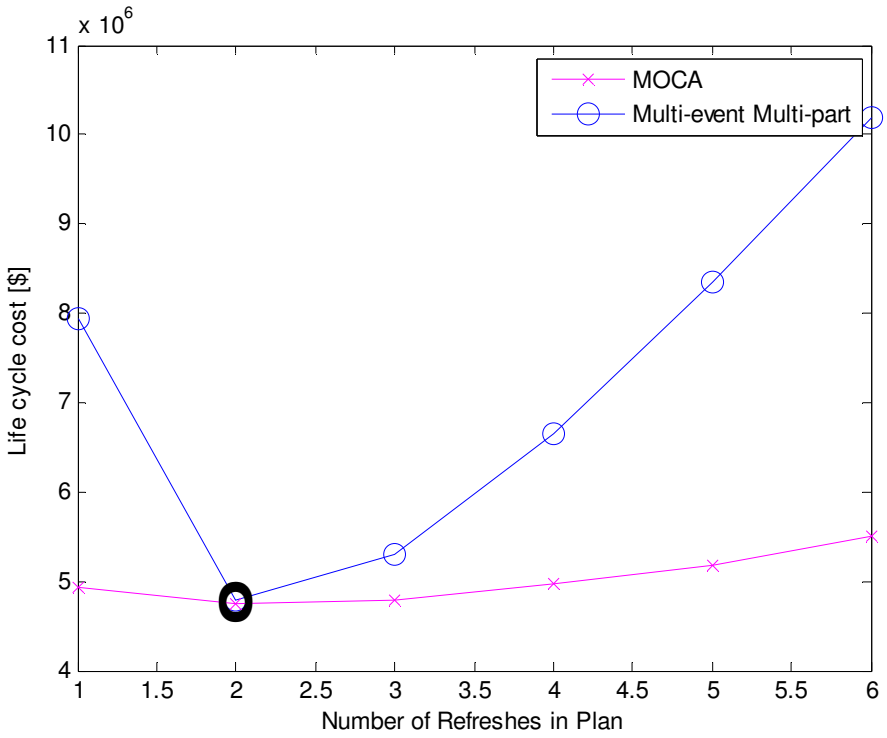


Figure 2-11: Case 3 results showing the optimal design refresh plans for both the MpMe and MOCA models using common parameters

Table 2-13 and Table 2-14 shows that the MpMe model's optimum design refresh plan is very similar to the MOCA model in both design refresh dates and the as we would expect the life-cycle cost. The first design refresh date was the same for both models. The MpMe second design refresh date was off by 9 years. This is an interesting result; however, this is only a single, specific case and thus generalizations cannot be made except that for this case, the MpMe model is comparable to the MOCA solution.

Table 2-13: Case 3 results from the MpMe model showing the lowest associated life-cycle cost for a given number of design refreshes per refresh plan.

	Optimal MpMe Design Refresh Plans, $\bar{x}$												TC_O[$]
N	0	1.8	4.5	9	12.6	18	23.4	27	31.5	34.2	36	45	
1												45	7.93E+06
2						***18***						***45***	***4.79E+06***
3				9				27				45	5.29E+06
4			4.5			18			31.5			45	6.66E+06
5		1.8			12.6		23.4			34.2		45	8.34E+06
6	0			9		18		27			36	45	1.02E+07

Table 2-14: Case 3 results from the MOCA model showing the lowest associated life-cycle cost for a given number of design refreshes per refresh plan.

	Optimal MOCA Design Refresh Plans, $\bar{x}$												TC_O[$]
N	0	1.8	4.5	9	12.6	18	23.4	27	31.5	34.2	36	45	
1						18							4.93E+06
2						***18***					***36***		***4.74E+06***
3						18					36	45	4.80E+06
4						18				34.2	36	45	4.97E+06
5						18			31.5	34.2	36	45	5.18E+06
6					12.6			27	31.5	34.2	36	45	5.50E+06

Case 3: Second Half – Parameters Unique to MOCA are Used

In this second half of case 3, MOCA will now be allowed to use all of its parameters set for this actual example system. Running the MpMe model again is redundant since none of its input data is changed.

Figure 2-12 shows the effect of including MOCA's additional parameters. As we would expect the mean dates for all the design refresh plans have not changed, but all the life-cycle costs associated with the design refresh dates are shifted upwards. Figure 2-10 and

Figure 2-12 vertical scales are in the same magnitude, but their axis limits are not the same. Regardless, it can been seen that MOCA's overall life-cycle costs were shifted upward.

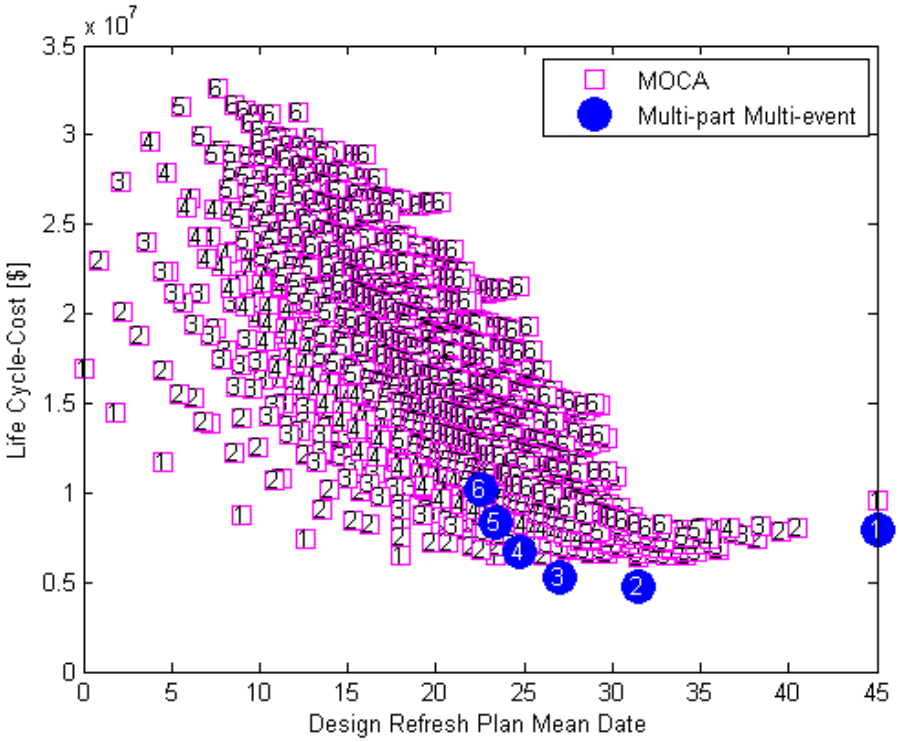


Figure 2-12: Case 3 results from MpMe and MOCA models using common and unique parameters

Figure 2-11 and Figure 2-13 show the upward shift effect more prominently since both vertical axes are in the same scale and have the same limits. Notice that despite this shift, the MOCA model finds the exact same optimal design refresh plan. This is an important result since it means that while the relative differences between the MOCA design refresh plans are closer to what they actually are using the additional MOCA parameters, the optimal design refresh plan remains the same.

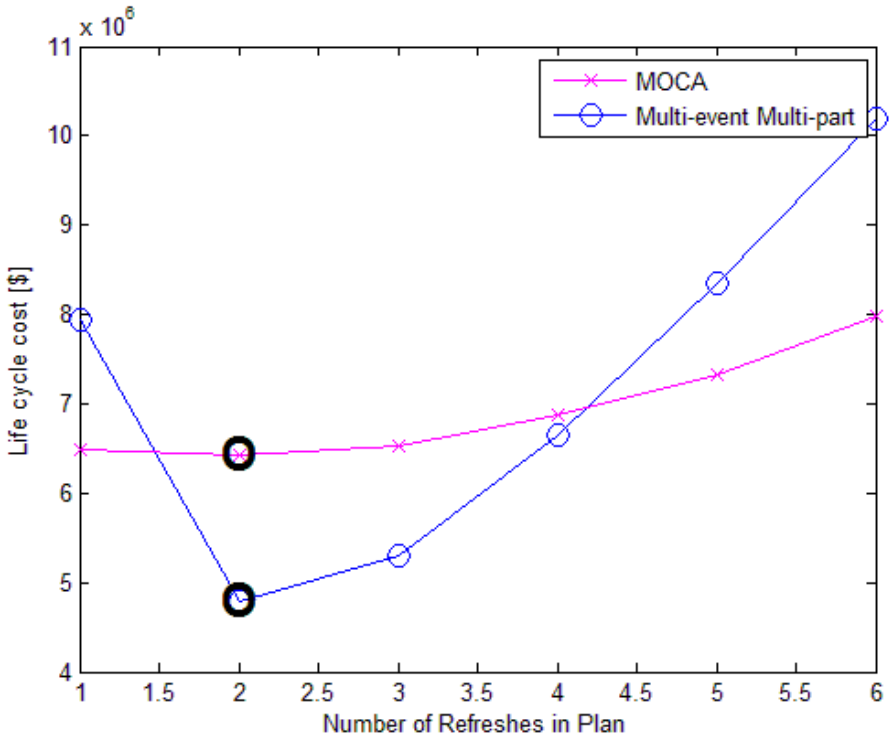


Figure 2-13: Case 3 results showing the optimal design refresh plans for both the MpMe and MOCA models using common and unique parameters

Table 2-15 is unremarkable and is not different from Table 2-13. Table 2-16 is different in some of the design refresh plans; however, the observation of note is that the optimal design refresh plan remains the same from the first half of case 3.

Table 2-15: Case 3 results from the MpMe model showing the lowest associated life-cycle cost for a given number of design refreshes per refresh plan.

	Optimal MpMe Design Refresh Plans, $\bar{x}$												TC_O[$]
N	0	1.8	4.5	9	12.6	18	23.4	27	31.5	34.2	36	45	
1												45	7.93E+06
2						***18***						***45***	***4.79E+06***
3				9				27				45	5.29E+06
4			4.5			18			31.5			45	6.66E+06
5		1.8			12.6		23.4			34.2		45	8.34E+06
6	0			9		18		27			36	45	1.02E+07

Table 2-16: Case 3 results from the MOCA model showing the lowest associated life-cycle cost for a given number of design refreshes per refresh plan.

	Optimal MOCA Design Refresh Plans, $\bar{x}$												TC_O[$]
N	0	1.8	4.5	9	12.6	18	23.4	27	31.5	34.2	36	45	
1							23.4						6.49E+06
2						***18***					***36***		***6.42E+06***
3						18					36	45	6.54E+06
4						18				34.2	36	45	6.88E+06
5						18			31.5	34.2	36	45	7.32E+06
6						18		27	31.5	34.2	36	45	7.99E+06

2.4.6 Case 3: Discussion on Solution Uniqueness

Solution uniqueness in the context of this dissertation is determining whether there exists one solution for every unique associated life cycle cost. It is possible to have two design refreshes different both in the number of refreshes and the refresh dates that have the same associated life cycle cost. This idea is evident from Table 2-17 which shows the convex behavior of both the MOCA and MpMe cost model functions. In general, the MOCA model has shown that it is capable of finding design refresh plans that have approximately the same associated life cycle cost up to the 4^{th} decimal place and this can be seen from Table 2-18 which shows some selected design refresh plans taken from the global optimum example in case 2. The two refresh plans that are approximately the same in associated life cycle cost are the design refresh plans with 3 and 6 refreshes in them.

Table 2-17: Selected Design Refresh Plans from the MOCA model

	Date(s) of Design Refresh(es) within MOCA Plan, $\bar{x}$												Lifecycle Cost, TC_O[$]
Number of refreshes within plan, N	**0**	**2**	**5**	**10**	**14**	**20**	**26**	**30**	**35**	**38**	**40**	**50**	
1									35				74.8698951881088
2	0						26						73.3499520469201
3		**2**								**38**	**40**		**73.4589597814505**
4			5	10		20				38			73.4251455433501
5	0					20			35		40	50	73.4961948133850
6			**5**	**10**				**30**		**38**	**40**	**50**	**73.4589225196813**

As for the MpMe model, it is less capable of finding a pair of design refresh plans that have an associated life cycle cost close in value since it evaluates only periodic refresh plans. Table 2-18 shows the 6 (and only 6) refresh plans produced by the MpMe model from the example given in case 2 and as noted all refresh plans are significantly different in associated life cycle costs.

Table 2-18: Design Refresh Plans from the MpMe model

	Date(s) of Design Refresh(es) within MOCA Plan, $\bar{x}$												Lifecycle Cost, TC_O [$]
Number of refreshes within plan, N	**0**	**2**	**5**	**10**	**14**	**20**	**26**	**30**	**35**	**38**	**40**	**50**	
1												50	101.121003355595
2						20						50	56.9214263909045
3				10				30				50	54.2688416286314
4			5			20			35			50	61.0822131225044
5		2			14		26			38		50	71.3793908135603
6	0			10		20		30			40	50	83.3142425408159

2.4.7 Case 4: Gradient Based Search - Unconstrained

Previous cases have been using combinatorial exhaustive search methods to find the optimum design refresh plan; however, in cases 4 and 5 the optimization will be done using a gradient based search method called the interior-point method. The mathematic programming language used to implement this optimization example is Matlab Version 7.12.0.635 (R2011A) which has a variety of built-in optimization methods. The Matlab optimization function used in this example is called fmincon which is composed of several optimization algorithms. As an option, the fmincon function can be forced to use a specific optimization algorithm. The interior point algorithm was selected to optimize the MpMe based on its efficiency to solve the example problem. Not within the scope of this dissertation, there is another category of optimization called the genetic algorithm (GA) that may prove to work better in finding the optimum then the two methods presented here. Genetic algorithm based optimization searches for the optimum by utilizing the theory that if two designs produce favorable results, then the combination of the two designs into a hybrid design is likely to produce an even better result. This idea of combining and rearranging various designs into other hybrid designs does play into the architecture of some DRP models since combinatorial exhaustive search methods essentially do the same thing as GA except they do all possible combinations rather than systematically iterating using an algorithm for the more favorable combination.

Using the same input data given for case 1 and case 2 (i.e., Table 2-5 and Table 2-6) the MpMe cost model problem is formulated as follows:

Find Y_R and N to minimize TC_O

$$\text{Min } TC_O(Y_R, N)$$

$$= PQY_R + PQ(Y_R + p_{max} - p_1)(1+r)^{(p_{max}-p_1)} \frac{1-(1+r)^{-((N-1)(Y_R+p_{max}))}}{(1+r)^{-(Y_R+p_{max})}-1}$$

$$+ \frac{PQ(p_{max}-p_1)}{(1+r)^{-(N(Y_R+p_{max})+p_1)}}$$

$$+PQY_R + PQ(Y_R + p_{max} - p_2)(1+r)^{(p_{max}-p_2)} \frac{1-(1+r)^{-((N-1)(Y_R+p_{max}))}}{(1+r)^{-(Y_R+p_{max})}-1} \quad \textbf{Equation 2-18}$$

$$+ \frac{PQ(p_{max}-p_2)}{(1+r)^{-(N(Y_R+p_{max})+p_2)}}$$

$$+C\frac{1-(1+r)^{-(N(Y_R+p_{max}))}}{1-(1+r)^{-(Y_R+p_{max})}}(1+r)^{-Y_R}$$

The intent of case 4 is to estimate the performance of the MpMe model while being optimized using a gradient based search, which in this case is the interior-point method. The MpMe model is written in a symbolic, closed form so that first and second derivatives can be derived for use in these types of gradient-based search methods. As always, the MpMe models needs a benchmark to compare to in order to establish computational advantages if any exist for the MpMe model. The benchmark for cases 4 and 5 will be the MOCA model and it will also be implemented for optimization using the interior-point method. To clarify what is being optimize, the MpMe model, under the parameters given for this case has only one variable that is controlled by the end of support equation (see Equation 2-17). The MOCA model is allowed to use *N* number of design refreshes per refresh plan that it can shift around to find the optimum design refresh plan under the conditions that no two design refreshes can have the same date, all refresh values must be non-negative and all refreshes must take place before the end of support date (*EOS*). Figure 2-14: Case 4 gradient based search progress graphFigure 2-14 plots the result of optimizing both design refresh models.

Notice that since the MpMe model has only one variable optimized (which represents all N refreshes within the refresh plan), it solves much faster than the MOCA model. As expected, the MOCA does find a better design refresh plan.

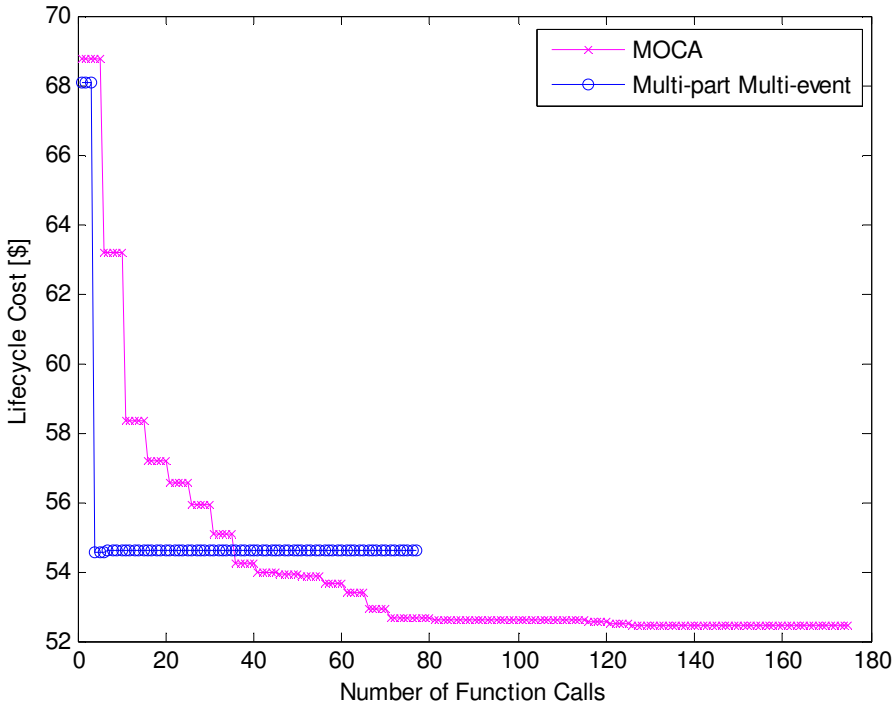


Figure 2-14: Case 4 gradient based search progress graph

Table 2-19 is a summary of the case 4 results for both models. Notice that both model were given the same initial refresh plan ($\overline{xo} = [xo_1 \quad \dots \quad xo_N]$). Another point of interest is the "Error" column which shows that the MpMe model has a 3.6% error but is over 50% faster than MOCA.

Table 2-19: Summary of MpMe and MOCA performance under the interior-point method

	MpMe	MOCA	Error
$\mathbf{xo_1}$	5	5	N/A
$\mathbf{xo_2}$	20	20	N/A
$\mathbf{xo_3}$	35	35	N/A
$\mathbf{xo_4}$	50	50	N/A
$\mathbf{Y_R}$	29.99	N/A	N/A
N	4	N/A	N/A
$\mathbf{C_{Total}}$	54.62	52.70	3.64%
$\boldsymbol{PSTD(\bar{x})}$	0	0.8683	N/A
$\mathbf{x_1}$	19.99	15.19	N/A
$\mathbf{x_2}$	49.99	37.64	N/A
$\mathbf{x_3}$	79.99	60.26	N/A
$\mathbf{x_4}$	109.99	84.29	N/A
No. of function evaluations	77	175	-56.00%
No. of iterations	19	34	-44.12%

2.4.8 Case 4: Discussion on Convexity

Introducing gradient-base optimization also brings into question where the MpMe model is valid for optimization, meaning for what conditions can the MpMe model truly be optimized? This section attempts to get an approximate idea of when the MpMe model can be truly optimized and when optimization is no longer optimization but just taking the lowest value of a monotonically decreasing function. Function convexity is a good indicator of when a function can be optimized. The conditions for a local minimum are defined as follows:

If is twice continuously differentiable and the domain is real, then we can characterize it as follows [43]:

$\dot{f} = 0$ First-order necessary condition (stationary point, local minimum or maximum)

$\ddot{f} \geq 0$ Secondary-order necessary condition for a local minimum

Using the ideas presented above, Equation 2-18 was twice differencitated and surface plots were created to get an overview picture of where the MpMe model is convex. Figure 2-15 is a plot of just the MpMe model with respect to Y_R and the non-dimentional term $\frac{P_0}{C_{DR_0}}$ which is a ratio of the component price over the cost of a design refresh. Notice from Figure 2-15 that for a low component price to high design refresh cost ratio, there seems to be no convexity. A high component price to high design refresh cost ratio produces what appears to be a convex MpMe model.

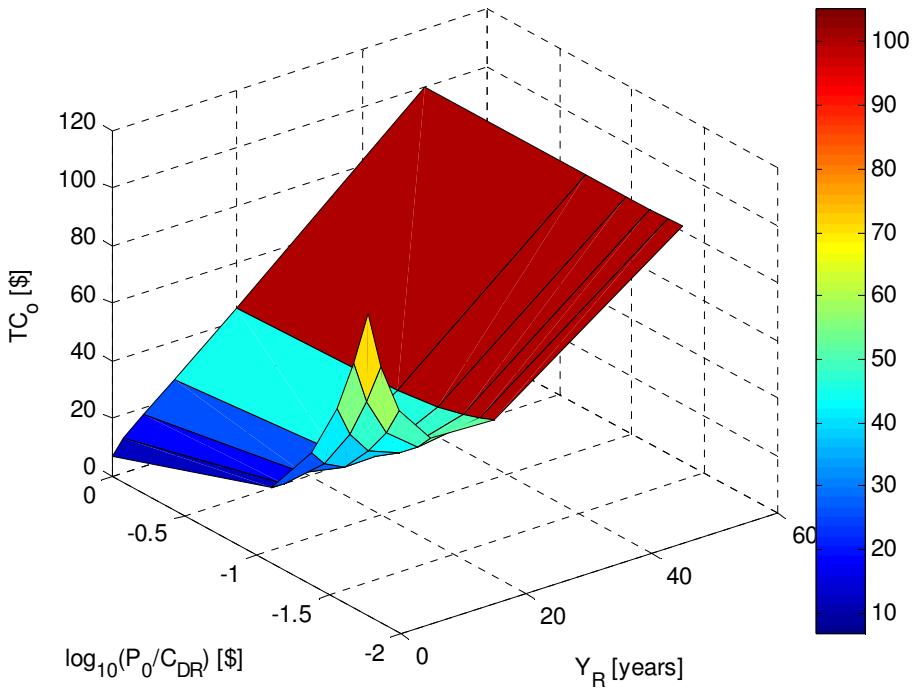


Figure 2-15: Surface plot of the MpMe model

Figure 2-16 is a surface plot of the first derivative of the MpMe model with respect to Y_R variable. According to the conditions defining a convex function, the same result is found looking at Figure 2-16, where on the plot, a green color indicates values larger than 0 and a color of red indicates a negative value. The boundary between the green and red region suggests a local optimum is achieved.

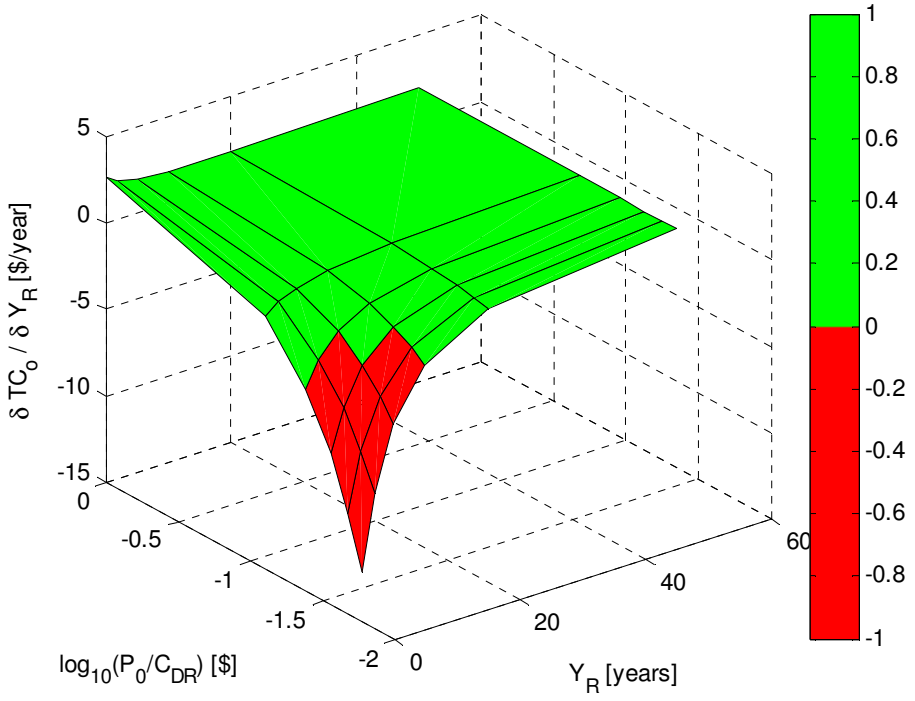


Figure 2-16:Surface plot of the first derivative of the MpMe model with respect to Y_R

Figure 2-17 is a surface plot of the second derivative of the MpMe model with respect to the Y_R variable. The region that produces a convex MpMe model is where the second derivative is positive or equal to zero. This reaffirms what was already said for Figure 2-15.

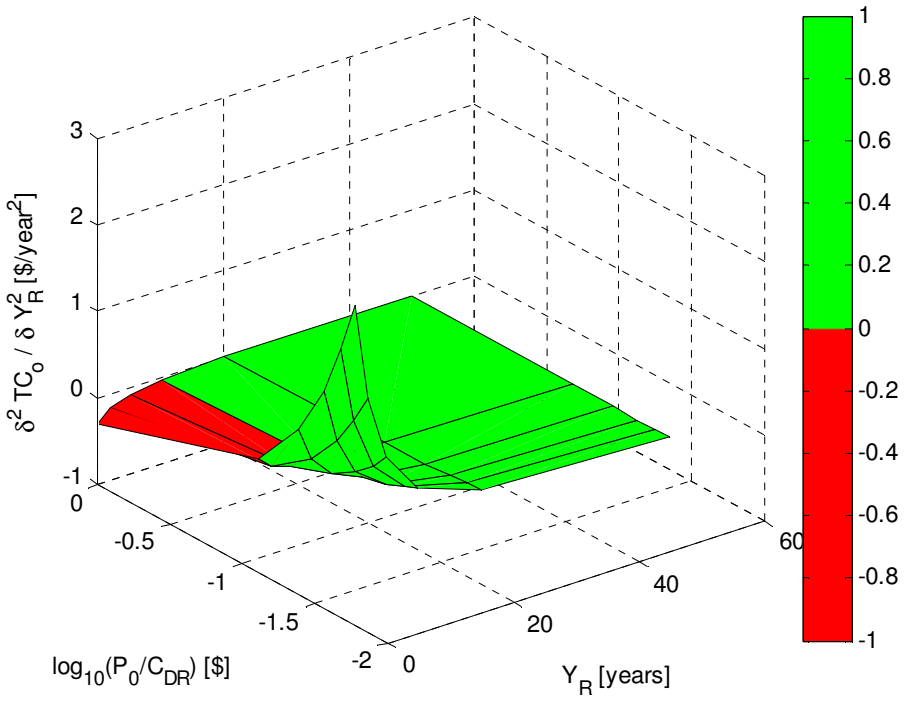


Figure 2-17: Surface plot of the second derivative of the MpMe model with respect to YR

2.4.9 Case 5: Gradient Based Search - Constrained

For case 5, the MpMe model is optimized with two temporal constraints (inequality constraints) imposed. The two temporal constraints are made up for this case study. The formulation of the MpMe model for case 5 is as follows:

Min $TC_O(Y_R, N)$

subject to:

$$g_k(Y_R, N) = 1 - \sum_{n=1}^{N} \sum_{k=1}^{K} \chi_k(Y_R, n) \leq 0$$ **Equation 2-19**

where Y_R and N are the design refresh wait time and the number of design refresh periods respectively. χ is the constraint period function that determines if any design refreshes are contained within the constraint boundaries defined by a_k and b_k. Below function χ is presented for review.

$$\chi_k(Y_R, n) = \begin{cases} 1, & a_k \leq nY_R + (n-1)p_{max} \leq b_k \\ 0, & nY_R + (n-1)p_{max} < a_k \\ & or \\ & b_k < nY_R + (n-1)p_{max} \end{cases}$$

Here we assume that $a_k < b_k$ for all k constraint periods. Table 2-20 shows the bounds of the temporal constraints.

Table 2-20: Temporal constraint bounds

Constraint Period, k	Lower Bound, a_k [Yr]	Upper Bound, b_k [Yr]
1	10	30
2	40	60

Figure 2-18 shows the optimization process of the interior-point method for both the MpMe and MOCA models with imposed temporal constraints. What is most notable about this graph is the fact that both models have about the same optimum life-cycle cost and that the MpMe model still has a significant computational effort savings over the MOCA model.

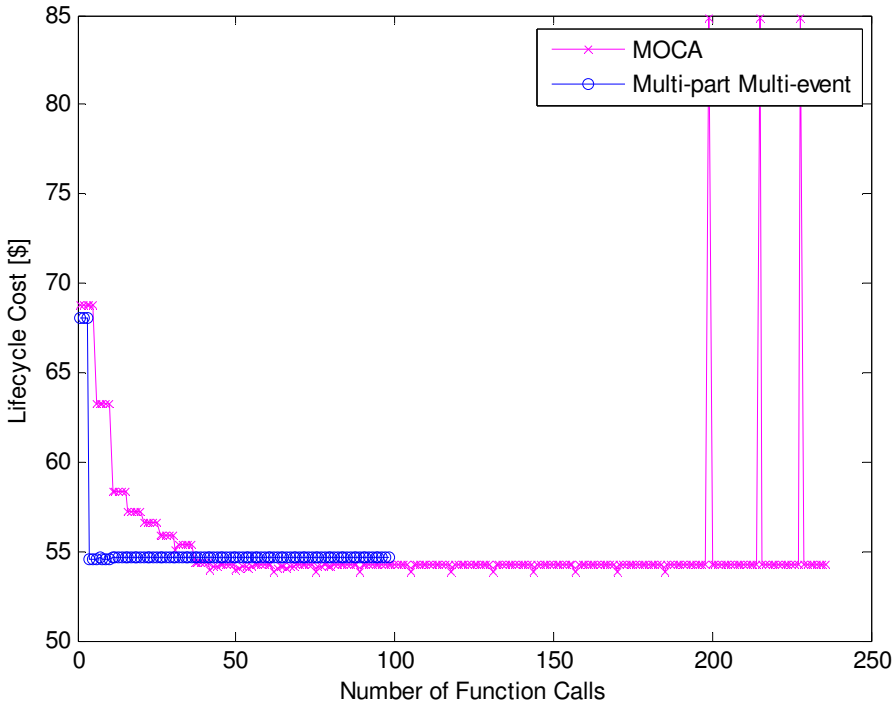


Figure 2-18: Case 5 gradient based search with constraints progress graph

The optimization programing used on the MpMe cost model was also implemented on the MOCA cost model as a comparison so that the advantages the MpMe cost model could be shown. Table 2-21 summarizes the final results of the optimized MpMe and MOCA cost models (Note: These results are local optimums. No scaling was done to the input data or design variables). There are a few points that need to be made before reviewing the results. The *xo* values are the initial design refresh dates that the optimization algorithm uses as a starting point to begin the optimization process. The *PSTD* function is defined in this dissertation as the periodicity metric and is a measure of the design refresh plan's periodicity. The *PSTD* function takes the periods of time between refreshes and finds the standard deviation of them. A periodic design refresh plan will have a *PSTD* value of 0 since all

periods between refreshes are the same. A non-periodic design refresh plan will have a *PSTD* value greater than 0. The *PSTD* function is defined as follows (Equation 2-20):

$$\bar{\mathrm{x}} = [x_1 \quad x_2 \quad x_3 \quad \dots \quad x_N]$$

$$T_n = x_{n+1} - x_n\,, \qquad n = 1, 2, 3, \dots, N-1$$

$$\mu T = \frac{1}{N-1}\sum_{n=1}^{N-1} T_n$$

$$PSTD = \sqrt{\frac{\sum_{n=1}^{N-1}(T_n - \mu T)^2}{N-2}} \qquad \textbf{Equation 2-20}$$

Table 2-21: Results from fmincon using interior point algorithm to optimize MpMe and MOCA models

	MpMe	**MOCA**	**Error**
$\mathbf{xo_1}$	5	5	N/A
$\mathbf{xo_2}$	20	20	N/A
$\mathbf{xo_3}$	35	35	N/A
$\mathbf{xo_4}$	50	50	N/A
$\mathbf{Y_R}$	29.99	N/A	N/A
N	4	N/A	N/A
$\mathbf{C_{Total}}$	54.62	54.21	0.77%
$\boldsymbol{PSTD(\bar{x})}$	0	6.8793	N/A
$\mathbf{x_1}$	19.99	16.00	N/A
$\mathbf{x_2}$	49.99	34.74	N/A
$\mathbf{x_3}$	79.99	39.77	N/A
$\mathbf{x_4}$	109.99	52.67	N/A
No. of function evaluations	98	236	-58.47%
No. of iterations	24	22	9.09%

According to the results presented in Table 2-1, the optimization algorithm found a local optimum with an error of 0.77% in life cycle cost than the local optimum found using the MOCA model; however, the MpMe uses 58% fewer function evaluations than the MOCA model but 9% more iterations than the MOCA model. This shows that even under temporally constrained conditions the MpMe model allow an optimization algorithm to find a comparable life-cycle cost that of the MOCA model and do it using less computational resources.

2.4.10 Case 5: Discussion on Data Required

By referring back to the DRP landscape chart shown in section 2.1, the MpMe model can now be plotted amongst the other DRP models. Notice that while the MpMe model as it is presented here in this dissertation does not have many effects included, but for the cases presented it has only slight deviation from the accepted life-cycle cost values produced from MOCA, while also doing this with less data and utilizing less computational effort. Looking at Figure 2-19 it can be estimated qualitatively that the MpMe model uses 50% less input data than MOCA model. While not in the scope of this dissertation, the amount of time needed to acquire all the data necessary to run a MOCA model will all of its parameters is significantly larger than that of the MpMe model.

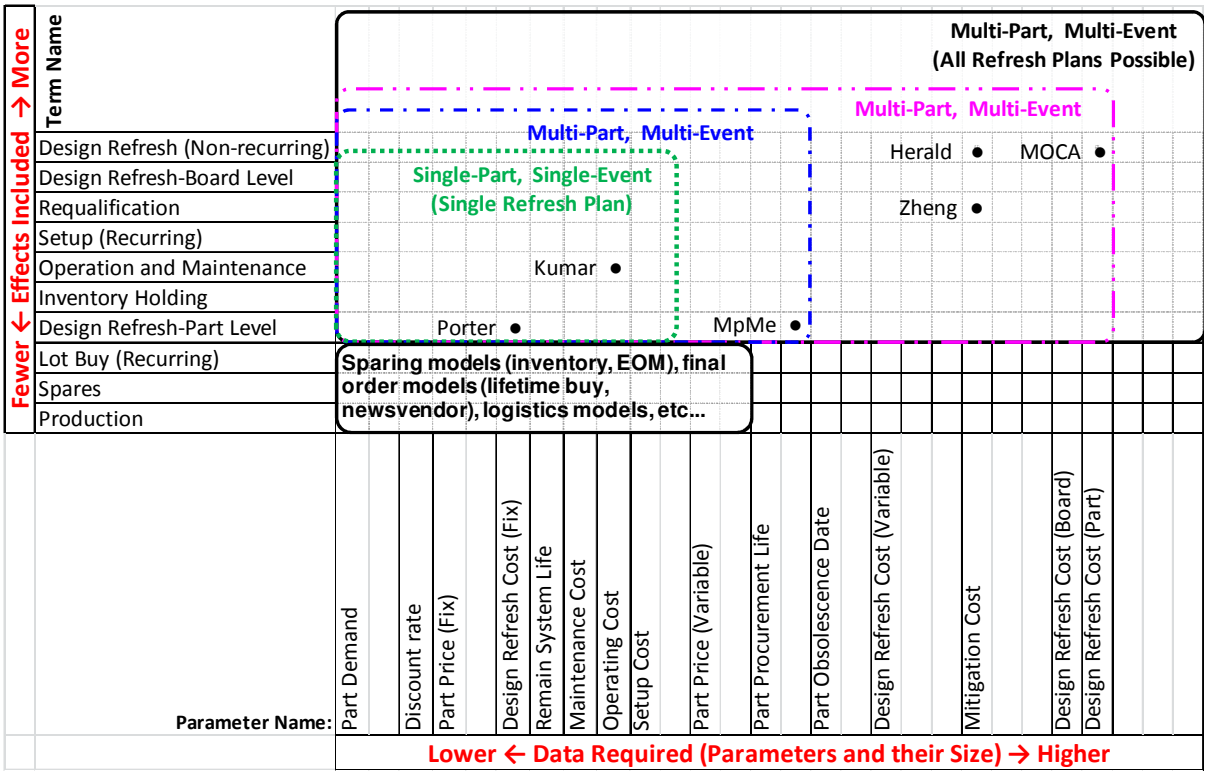


Figure 2-19: Qualitative comparison of data required between the MpMe model and other various design refresh planning models

2.5 Case Study Conclusions

The MpMe model has been verified against MOCA for the specific cases performed. For a problem that has a periodic refresh solution requirement, the MpMe has been demonstrated to:

- Be a closed-form model that can be twice differentiated (for at least a two component system)
- Use less input data than a more advanced DRP model capable of evaluating periodic and non-periodic design refresh plans
- Use less computational resources even with temporal constraints imposed

Additional conclusions that can be made from the case studies are shown below:

- MOCA produces the same life-cycle cost when evaluating the same refresh plan used in the MpMe model
- The MpMe model results in the same optimal number of design refreshes that are found using MOCA when only periodic refreshes are allowed
- The MOCA life-cycle costs are shifted up as a result of including additional parameters not modeled in the MpMe model; however, it did not affect the optimal number of design refreshes nor the optimal design refresh dates
- The MpMe model has over 50% functional call savings over using the MOCA model with negligible error in the life-cycle cost with and without temporal constraints imposed

Chapter: 3 Refresh Planning Constraint Formulation

Constraints imposed in the design refresh planning process can reflect technology roadmap requirements, obsolescence management realities, logistical restrictions, budget ceilings and management policy.

This chapter discusses the formation of constraints imposed on the management of a system by taking pre-existing knowledge of the system's limitations and translating it into an explicit form that can be directly applied to the system's design variable(s). The benefits of creating definitions that facilitate the translation of knowledge into constraints will be addressed and three general definitions are developed and used as a foundation to create constraints based on the obsolescence management policies of real world business organizations.

3.1 Constraint Taxonomy

The constraint taxonomy in this chapter was created from the view point of an organization sustaining a system. This organization can be a private company, government agency, or any group sustaining a system. Systems can range from desktop computer to aircraft. The term sustaining retains the same meaning as defined in Section 1.1, which essentially means to maintain existing systems and produce new systems. In the following relativistic words such as "I" or "internal" refer to that organization. Figure 3-1 shows a constraint taxonomy that classifies constraints based on the permission and ability to perform a design refresh activity (the taxonomy in Figure 3-1 is akin to the classic English lesson that differentiates "May I" and "Can I") [44].

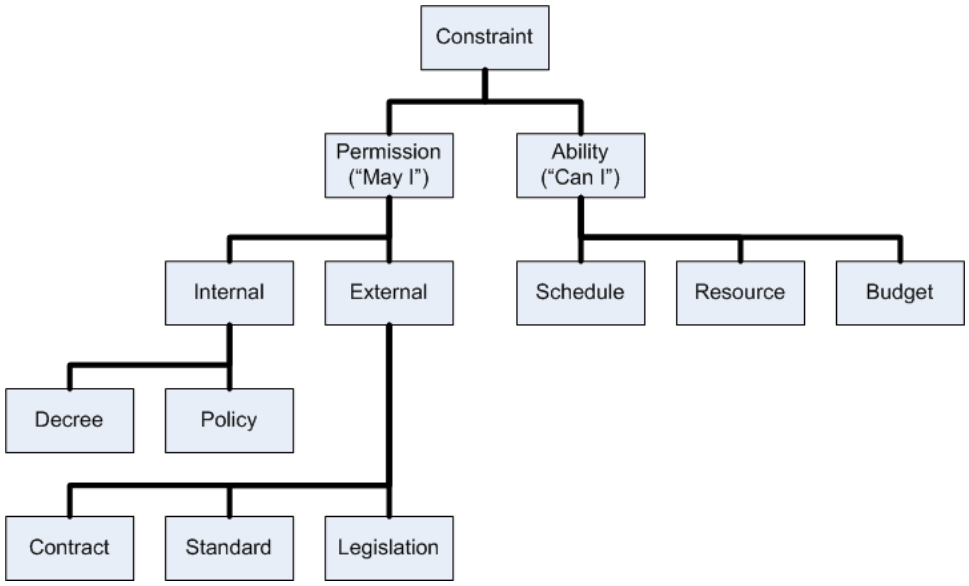


Figure 3-1: Constraint taxonomy from the view point of the design refresh planner.

A permission-based constraint represents a restriction imposed by an authority entity on the organization sustaining a system. The authority entity can be a single person or a group of people; however, the only stipulation that qualifies an authority figure is that whatever restrictions imposed by the authority figure must be obeyed by the organization sustaining a system otherwise penalties and other various enforcement activities on the organization sustaining a system will be incurred.

Permission-based constraints are separated based on whether a constraint is enforced externally or internally. For example, an externally imposed permission constraint could be a law enacted and enforced by the Federal government. An internally imposed permission-based constraint could be a company policy created by management and enforced by supervisors.

There are three categories of external sources of authority that can originate permission-based constraints: contract, legislation and standard. A contract is a legally binding agreement between the organization sustaining a system and a non-affiliated entity

such as a component supplier. Any contractual commitments between the two that affect the design refresh activities performed by the organization sustaining a system create permission-based constraints. A piece of legislation such as any law that affects the design refresh planning activities creates a permission-based constraint. A standard is a document that establishes a rule or measure (either minimum or optimum) for quality or level of performance. Any standards adhered by an organization in order to establish a certification of some kind is a source of permission-based constraints.

Next are the two internal sources of permission-based constraints: policy and decree. Policies are usually created over time as problematic issues arise that warrant an internal "design rule" or guideline used as a tool to mitigate or even avoid the problem from occurring. When a policy affects the planning of a design refresh it creates a constraint that is self-imposed. Decrees (i.e., an executive order) unlike policies do not necessarily encompass every aspect of the life-cycle management process; rather decrees should be viewed as policies limited to specified aspects of the life-cycle planning process and generally smaller in scope and complexity. The decree can be viewed as instantaneous restrictions to specific aspects of the design refresh planning activities, whereas policies encompass every aspect of the refresh planning process. Decrees and policies belong to the permission-based category of constraints because these constraints if broken do not physically prevent an organization who is sustaining a system from performing an activity, rather the actors within the organization who violate these permission-based constraints will suffer various disciplinary actions such as penalties and punishments resulting in a disruption of life-cycle management activities.

The ability to perform an activity is based on physical parameters such as the available funds, resources, and time. Permission to perform an activity is based on the authorization from imposed written law, supervisory actors (e.g., company management, business owner), applicable national/international standards and specifications, and contractual commitments. Ability-based constraints represent all the capacities of an organization for managing a system. There are three aspects in any organization that have quantifiable capacities: scheduling, budget and resources. Scheduling constraints can require certain activities such as design refresh activities to occur before a specified date. Budget constraints can place expenditure ceilings on all activities, preventing over spending. Resource constraints prevent using more than the available resources such as people or workspace.

In order to implement the various constraints shown in Figure 3-1, the ways in which a constraint can restrict DRP activities need to be established.

3.1.1 Ways to Restrict DRP Activities

Any constraint imposed on the DRP process will control the planning of a design refresh in three fundamental areas: financial, logistical and temporal. For example, a constraint that restricts financially may place an upper bound on the money available to perform design refreshes or other management activities in a particular period of time. A constraint that constrains logistically would restrict the number of facilities performing design refreshes (e.g., a finite number of dry docks for ships). A constraint that restricts temporally will require the design refresh activity to complete within a specific period of time for technology insertion to upgrade a system's capability, or may preclude specific

periods of time for design refresh because the system to be refreshed is unavailable (e.g., a submarine is gone for 12 months and the design refresh cannot be performed at sea).

This dissertation focuses on how constraints restrict temporally and specifically how they are implemented in the management of sustainment-dominated systems. For the DMSMS affected systems, constraints that restrict temporally are the most prevalent DRP drivers.

3.1.2 Convertibility of Constraint into Penalty Function

In optimization, constraints can be handled in different ways including transforming them into different forms such as inequalities to equalities using slack variables when performing linear programming (i.e., simplex method) [45] or converting the constraint into a penalty function which essentially adds a cost to the objective/cost function when a constraint is violated [46]. This allows both the constraint violation and objective function to be minimized concurrently; however, this is not always possible. The taxonomy presented here gives insight as to what constraint classifications can be converted into a penalty function. Since permission based constraints can be violated by paying a fee then permission based constraints can be converted into penalty functions. Ability based constraints cannot be violated since the constraints represent physical, quantitative limits when if violated the DRP process is assumed to no longer be able to complete its activities, which is considered a failure to sustain the system and is the reason why ability based constraints cannot be converted into a penalty function.

3.2 Constraint Formation

Constraints that restrict the timeline of a design refresh planning activity usually take the form of inclusive inequalities because they represent ranges of time when at least one design refresh is required to be completed. These constraints are typically derived from events that after their occurrence will affect the life-cycle management activities such as component obsolescence events, legislation enactment events, and the publication of new standards or certification events.

Temporal constraints that are usually bounded ranges of time that require one or more design refresh activities to complete within them; however, the bounds are unknown at the beginning of the DRP process. What is known at the beginning of the DRP process is the event that results in the temporally restrictive constraint. To determine the constraint bounds for constraints that are the result of a component's[8] obsolescence we must determine which of the following scenarios applies – does the component's obsolescence event: 1) affect both the operation and the production of the system, 2) affect production and not operation, or 3) not affect either activity. It will be assumed that if the component's obsolescence event affects the operation of the system, then the production of the system is also affected. Knowing how the component's obsolescence event affects the operation and production of a system will determine if and how an explicit constraint will be constructed. Three possible general scenarios called obsolescence event types have been identified, and their definitions are described in the next section.

[8] "Component" refers to the lowest management level possible for the system being analyzed. In some systems, the "components" are laptop computers, operating systems, and cables; while in other systems the components are integrated circuits (chips).

3.2.1 Obsolescence Event Type Definitions

In the following obsolescence event type definitions, the term obsolete can take on several meanings depending on the component restricted by the system constraint. If the component is a piece of hardware, obsolete generally means you cannot procure the item from the original manufacturer; however, in some cases the item may remain available from your existing inventory or from aftermarket sources. If the component is a single legal copy of software, obsolete usually means you can no longer obtain software updates such as service packages or security patches. If the component is a particular skill or capability, obsolete means that persons performing the skill or the process/equipment necessary to support the capability is not accessible.

3.2.1.1 "Weak" Obsolescence Event

No change to previously fielded (installed) systems or systems to be manufactured in the future is required. As long as the obsolete item is available (from existing stock or aftermarket sources), new systems can be manufactured and fielded using it and previously installed systems can be repaired with it if necessary.

System constraints often identify hardware (electronic components for the applications discussed in this dissertation) as being a Weak obsolescence event. The rationale behind this is that if hardware goes obsolete there is no reason to change it as long as you have access to a sufficient supply of the obsolete component to satisfy manufacturing and support requirements.

3.2.1.2 "Strong A" Obsolescence Event

Fielded (installed) systems can continue to operate with the obsolete item and can be repaired with the obsolete item if it needs replacement due to a failure of the item. However, new systems to be manufactured in the future cannot be built and fielded with the obsolete item (whether the obsolete item is available or not).

3.2.1.3 "Strong B" Obsolescence Event

Fielded (installed) systems are not allowed to continue to operate with the obsolete item and must be backfitted within a defined time period. New systems cannot be built and fielded with the obsolete item (whether the obsolete item is available or not).

An example of a system constraint that identifies a component's obsolescence event as "Strong B" is an electronic data security policy. Consider a military ship-board communication system that has computers on its network that are connected to the public web running a commercial operating system that is about to reach its end of support date (the effective obsolescence date for the software), end of support means the end of security patches and the potential for a security risk if not replaced. In this example the operating system is the component. To maintain its security integrity the customer for the system puts in place a policy that the computers cannot continue to operate with the obsolete operating system, so any installed systems with the obsolete operating system will have to be backfitted[9] and any new instances of the system will have to be delivered with a non-obsolete operating system.

[9] A backfit consists of a refresh of the fielded version(s) of the system and an implementation of the refresh on all fielded applicable instances of the system. The number of implementations of the backfit refresh is

3.2.2 Other Event Definitions

The idea of creating an event definition to eliminate the problems of interpreting a policy or other implicit limitations imposed on a system is not limited to the effect of obsolescence. Other event definitions can be constructed from component reliability, standards/requirements updates, and legislation (e.g., RoHS, WEEE, EPA emission standards) for example.

A component which experiences or causes failures that occur with a level of frequency (i.e., "bad actor") that is significantly higher than other components creates a requirement to design the component out of the system before manufacturing of future instances of the system and in fielded systems [47]. This requirement is actionable only when the "bad actor" is identified that can be accomplished after a root cause analysis (RCA) event occurs.

A system may have to satisfy scheduled re-certifications/inspections/performance based tests throughout its life cycle regardless of whether it is operating within the original design specifications or whether it has been changed since being fielded. An example of this is a factory that produces large quantities of CO_2 that must pass an emission test to determine if it is performing within the allowable CO_2 limits, otherwise penalty costs will be incurred. This is a "pay a penalty" or "perform a design refresh" scenario, which is different from the "Strong B" scenario where a design refresh must be performed. System design changes must be made in order for the factory to capture or eliminate the excess CO_2 or pay the penalties. This includes the factories in operation and future constructions of those factories [48].

---

determined by reviewing all fielded versions of the system and accumulating appropriate quantities of affected system elements.

The "bad actor" and legislation type of events have not been included as constraint creating events because they actually represent choices (with associated penalties), not policies.

The next section will explain the process of taking the information known about the system constraint (i.e., the obsolescence event type) and forming an explicit DRP constraint.

3.3 Constraint Formation Algorithm

Component instances that are in a "Strong" obsolescence event category will be examined because the "Strong" obsolescence events result in the creation of constraints imposed on the DRP process.[10] It is important to note that the obsolescence event types are not necessarily dependent on the component, but rather the relationship between the component and where it is located in the system (i.e., the component's context). A constraint may not specify an exact component, but rather a specific effect a component's obsolescence event has on the system. The same component could appear in multiple locations within a system and generate a different constraint in each case; therefore, every component must be examined in a system even if it is not unique.

This section presents an algorithm that generates (synthesizes) temporal constraints (i.e., constraints that constrain temporally). The inputs for this algorithm are the production/deployment schedule, the system bill of materials, forecasted obsolescence information on all components, end of support dates, the refresh plan under consideration and obsolescence mitigation assumptions such as look-ahead time, and replacement component assumed procurement life upon adoption. The numerical values, such as those that pertain to

[10] By definition, weak obsolescence events do not require a change to the system, thus no constraints are imposed on the DRP process as a result of a weak obsolescence event.

dates, can include uncertainties in this algorithm. This is important especially since the input uncertainty is often large for DRP problems.

3.3.1 Step 1: Synthesize Constraints Associated with "Strong" Events

In order to build temporal constraints for components that are identified as causing a "Strong" obsolescence event we need to determine the constraint start date (a_k) and the constraint end date (b_k), which form the constraint period.

To better illustrate the construction of a temporal constraint, a graphical representation of the constraint's construction will be presented (Figure 2-2). First a timeline representing time increasing from left to right is created and a "Strong" obsolescence event is placed on the timeline as a point in time. For both "Strong A" and "Strong B" obsolescence event types, each require a design refresh activity to be performed on or sometime before the obsolescence event occurs. It is possible to create an equality constraint that forces a design refresh activity to complete at a specific point in time; however, this limits the ability of a single design refresh activity to satisfying multiple constraints. Creating an inequality constraint will allow a single design refresh activity to complete within a range of time and potentially satisfy multiple constraints, but how far before the obsolescence event can the design refresh activity complete its activities and still satisfy the constraint? The answer lies in how a design refresh decides which obsolete components are to be refreshed. Before a design refresh activity can begin it is necessary to determine which components' obsolescence issues are to be resolved. The design refresh starts by looking backward in time from where the design refresh activity completes on the timeline and creates a list of components that have become obsolete since the last design refresh completed or since the system was first built – this list of obsolete components are replaced at the design refresh. In

addition, the design refresh can utilize forecasted obsolescence information about the system's components and "look ahead" in time to see which components are likely to go obsolete and proactively replace them at the refresh before they are obsolete. Since interest is placed on when a design refresh completes its activities rather than when it starts, the "look ahead" time is measured from when the design refresh activity completes. The "look ahead" time is a parameter that does not change throughout the DRP process. This idea of a "look ahead" time is used to determine the start of the temporal constraint period.

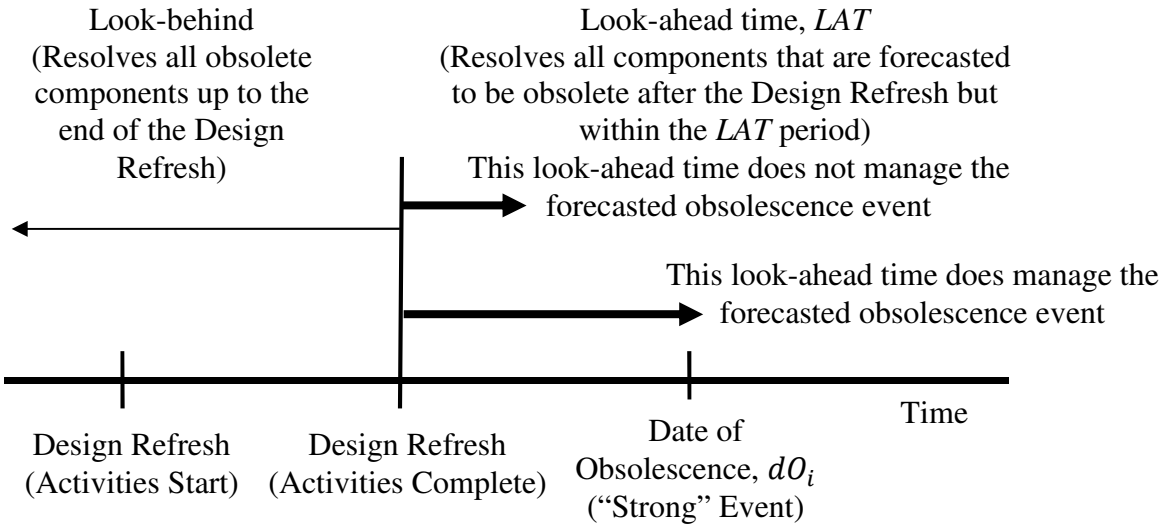


Figure 3-2: The determination of which obsolescence events to resolve at a design refresh.

The constraint start date is calculated by subtracting the look-ahead time (*LAT*) from the forecasted date of obsolescence (dO_i), see Equation 3-1 and Figure 3-3. The definition of a look-ahead time is the amount of time the refresh plan "looks ahead" during a design refresh for forecasted component obsolescence issues and pro-actively removes components

that are forecasted to have obsolescence problems within the *LAT* of the completion of the current design refresh activity.

$$a_k = dO_i - LAT \quad \textbf{Equation 3-1}$$

The *LAT* is limited by how far into the future the obsolescence forecasting method can forecast and is set based on how it affects the life cycle cost of the system and potentially on how it affects the satisfying of constraints.

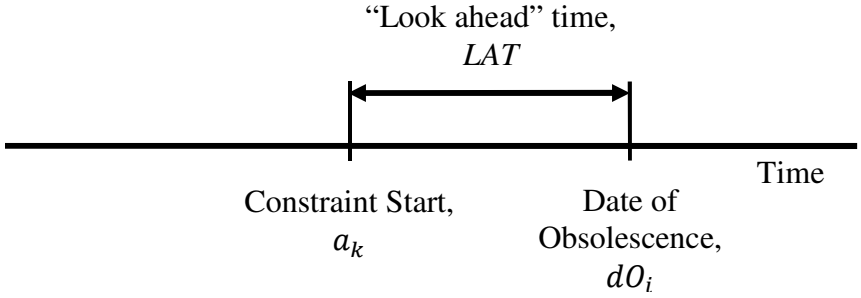


Figure 3-3: The relationship between "look ahead" time, the constraint start, and the forecasted date of obsolescence.

Equation 3-1 determines the constraint start date (a_k), which will be used to create a lower bound constraint that will be paired with an upper bound constraint whose combined effect will constrain the design refresh to occur within the bounded range of time. This lower bound constraint is formed first by making the statement: "The design refresh must occur on or after the constraint start date (a_k)," which is represented as:

$$a_k \leq x_n \quad \textbf{Equation 3-2}$$

The variable *x* in Equation 3-2 is one design refresh date out of *r* design refresh dates placed in a vector that make up a design refresh plan ($\bar{x}$). Rewriting Equation 3-2 in a general form,

$$\overrightarrow{g_1}(\bar{x}) = a_1 - x_n \leq 0 \quad \textbf{Equation 3-3}$$

The constraint end date (b_k) depends on the type of "Strong" obsolescence event. In the case of a "Strong A" obsolescence event, the constraint end date is the next date of production (dD_m), also called a production event, i.e., the next date when the component is needed to support the system (manufacturing or sparing). A production event includes all the activities that result in the creation of a system instance or the replenishment of spares. The amount of time between the a_k and b_k consists of two periods: the look-ahead time and the "waiting time" (Y_{R_k}). The "waiting time" is specific only to "Strong A" constraints and is the time the component is allowed to remain obsolete within the current system design after which a design refresh must occur. This secondary period of time is called "waiting time" because the component is "waiting" for a design refresh after it has gone obsolete.

$$Y_{R_k} = b_k - dO_i = dD_m - dO_i \qquad \textbf{Equation 3-4}$$

The "waiting time" was defined to distinguish between the constraints created from the "Strong A" and "Strong B" obsolescence types (Figure 3-4). It is not a parameter that can be assigned a value; rather it is a measure of time between the obsolescence event of a component and the production event that follows it.

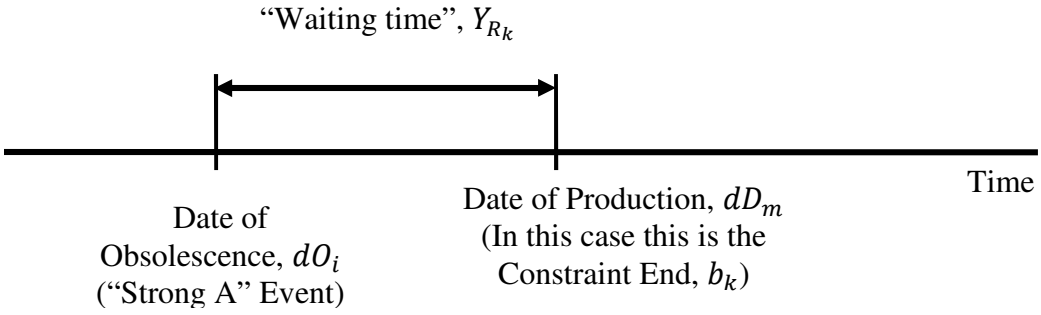


Figure 3-4: The relationship between "waiting" time, the forecasted date of component obsolescence, and the next date of system production.

In the case of "Strong B" obsolescence event types, an immediate design refresh corresponding to the obsolescence event is required. Just like the "Strong A" constraints,

"Strong B" constraints have a period of time before the obsolescence event called the look-ahead time and Equation 3-1 can be used to find the constraint start date (a_k). Unlike "Strong A" constraints, "Strong B" constraints do not have a "waiting time" because by definition they require an immediate design refresh, so the constraint end date (b_k) is the same as the obsolescence date (dO_i), (i.e., $b_k = dO_i$). With the constraint end date known, the upper bound constraint can be formed by making the statement: "The design refresh must occur on or before the constraint end date (b_k)." This is represented as:

$$x_n \leq b_k \qquad \textbf{Equation 3-5}$$

Rewriting Equation 3-5 in a general form,

$$\overleftarrow{g_1}(\bar{x}) = x_n - b_1 \leq 0 \qquad \textbf{Equation 3-6}$$

Equations 3-3 and 3-6 form an overlap of inequality constraints that result in a bounded range. This range is the constraint period during which a design refresh (x_n) must complete its activities.

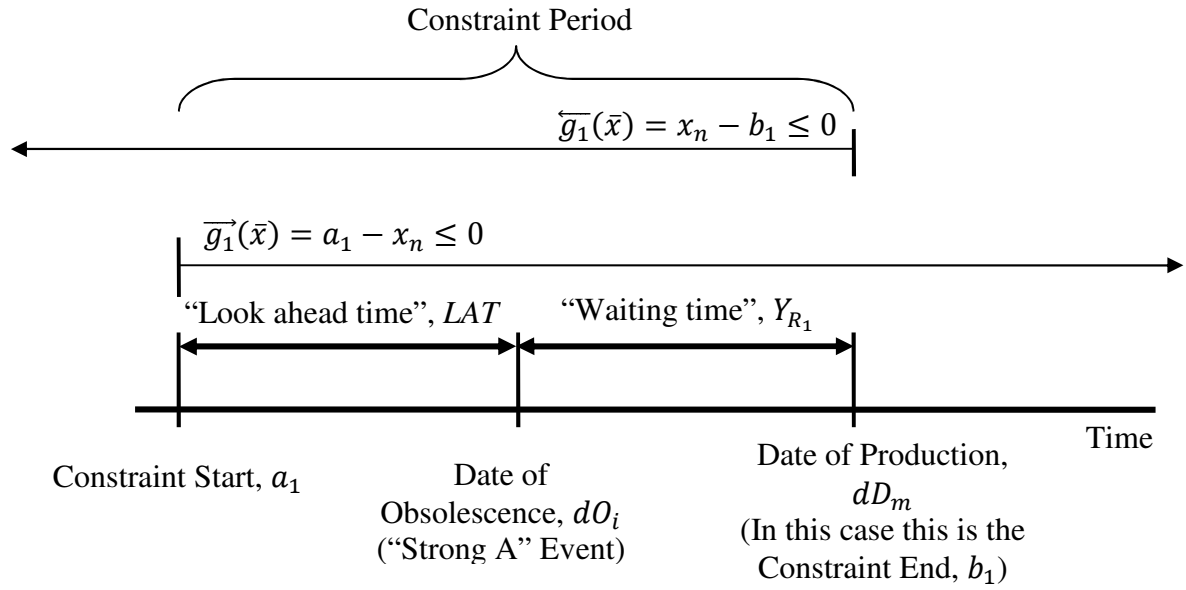


Figure 3-5: Graphical construction of the "Strong A" temporal constraint.

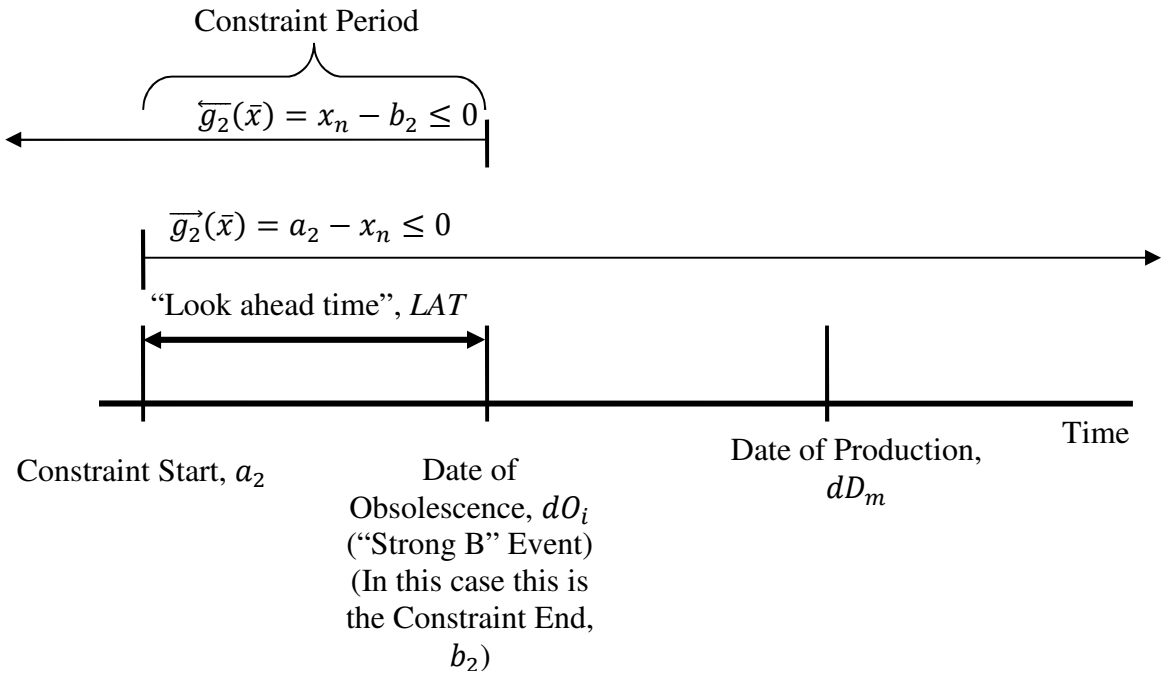


Figure 3-6: Graphical construction of the "Strong B" temporal constraint.

The biggest difference between "Strong B" and "Strong A" constraints is that "Strong B" constraints include a backfit for all fielded systems that are affected by the obsolescence of the "Strong B" component since components that create a "Strong B" obsolescence constraint cannot remain in any fielded instance of the system. Once a design refresh has designed out the soon to be obsolete component, the design refresh is implemented in all fielded instances of the system which is known as backfitting. The cost of the backfitting process can be broken into two parts: the backfit development cost (a non-recurring cost) and the backfit implementation cost (a recurring cost for each fielded system instance). To implement the backfit, an additional production date is inserted into the production schedule at the same date of the "Strong B" obsolescence date so that costs associated with the implementation such as inventory and carrying costs are reduced, otherwise known as a just-in-time refresh strategy (see footnote 19 in Case Study, Section 5.1). Similar to a production

event that produces new instances of the system, this inserted production event has a production quantity that is the number of affected, fielded system instances; however, in this context it is implementing backfits rather than creating new system instances. This inserted production event (i.e., backfit implementation event) is treated the same as any other production event. This inserted production event should not be used as a constraint end point when constructing "Strong A" based constraints since by definition components that create a "Strong A" obsolescence event can remain constituent components as long as no new system instances are being produced. And since this inserted production event is not producing new system instances, any obsolete "Strong A" component can remain obsolete until the next production event that produces new instances of the system.

3.3.2 Step 2: Creation of Component Replacements and Their Constraints

In many cases the procurement lifetimes of electronic components are significantly shorter than the manufacturing and support lives of the sustainment-dominated systems they are in, therefore, a component's replacement (at a design refresh) may also go obsolete before the support of the system terminates [49]. In order to model this effect, the components that replace the original component also need forecasted procurement lifetimes and obsolescence dates. This means that additional constraints associated with the synthesized replacement components also need to be created.

This step does three things: inserts a replacement for the predecessor component[11] that went obsolete; generates the replacement component's obsolescence date; and if the

[11] The predecessor component is a component followed or replaced by another component (the replacement component).

replacement component's obsolescence event is before the end of support date of the system, it must create additional replacement components.

In order to determine whether the simulated replacement component will go obsolete within the analysis period, the obsolescence date of the replacement component must be generated. The three pieces of information needed to model the replacement component's obsolescence date are the procurement lifetime [49], the life-cycle code of the replacement component, and the obsolescence date of the predecessor component. For simplicity, assume that the procurement lifetime, the length of time the component can be procured from its original manufacturer, of the replacement component is the same as the predecessor component. Next, the life-cycle code of the replacement component is selected. Depending on the application (i.e., risk tolerance for the adoption of new components) different component maturities could be targeted. A component's maturity is defined by where it is on its life-cycle curve at a specific point in time [50]. The life-cycle curve is divided into regions that reflect the component's maturity that correspond to the following life-cycle codes: 1 = emerging, 2 = growth, 3 = maturity, 4 = decline, 5 = phase out, 6 = obsolete.

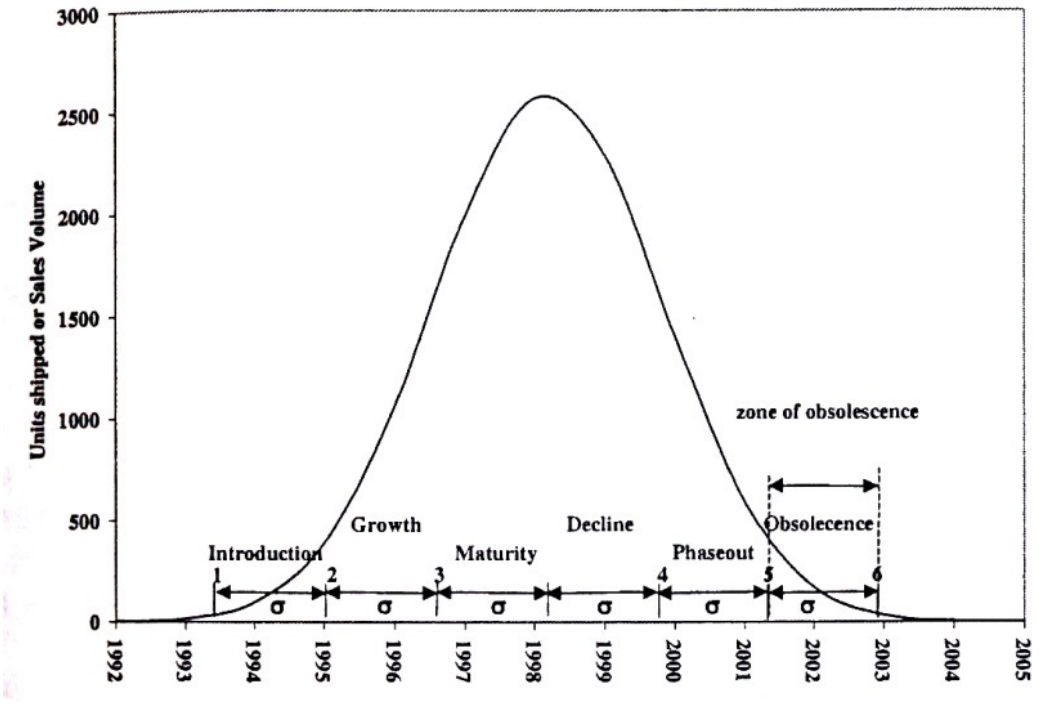


Figure 3-7: Product life cycle curve [32].

Sustainment-dominated systems are usually extremely risk adverse and may only select components that have life-cycle codes of 2 or 3 (whereas a high-volume commercial application might choose components with life-cycle codes of 1 because their success depends on being state-of-the-art). With the procurement lifetime and life-cycle code selected, the obsolescence date for the replacement component can be calculated. Equation 3-7 is used to generate the obsolescence date of new components introduced at design refreshes (see Figure 3-8).

$$D_o = D_{pc} + L\left[\frac{I_o - I_R}{I_o - I_I}\right] \qquad \textbf{Equation 3-7}$$

where

D_o = Date of obsolescence
D_{pc} = Date of obsolescence for the predecessor component
I_o = Life-cycle code indicating component is obsolete
I_I = Life-cycle code indicating component is emerging

I_R = Life-cycle code of synthesized replacement component
L = Procurement lifetime

In the event that the procurement lifetime of the original component is not known or cannot be determined, then the procurement lifetime corresponding to the component type can be used. The procurement lifetime of the component type is the average of the lifetimes of all the components in the system's bill of materials that have the same functional type. Life codes, obsolescence dates and procurement lives for existing electronic piece parts can also be obtained from commercial electronic component databases including: Silicon Expert, Total Parts Plus, HIS Parts Universe, Q-Star, and Part Miner [51], [52], [53], [54], [55].

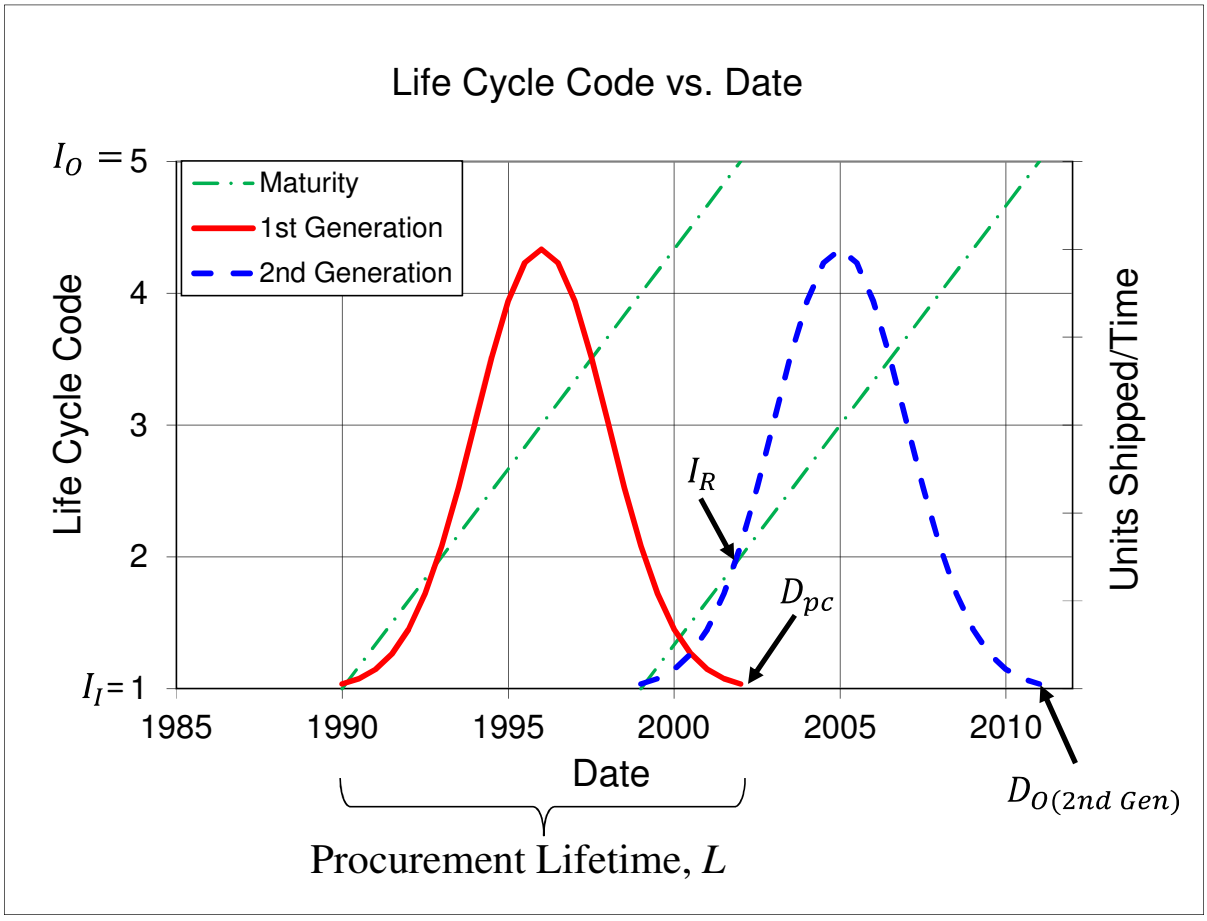


Figure 3-8: Graphical representation of predecessor component life-cycle curve (solid line) overlapping with the synthesized replacement component at a beginning of the "growth" phase (i.e., I_R = 2) of the replacement component's life-cycle (dash line).

Once an obsolescence date for the synthesized replacement component has been determined, if it is earlier than the system's end of support date, then the type of obsolescence event associated with the component will determine how the calculated obsolescence date is used to generate subsequent constraints.

In the case of a "Strong A" constraint, the obsolescence date for the synthesized replacement component must be later than previously created constraint end date for the predecessor "Strong A" component since the predecessor "Strong A" component's obsolescence event does not force a design refresh for the system until the next production event. If the obsolescence date of the synthesized replacement component is not later than the previously created predecessor "Strong A" component constraint end date then the procurement lifetime of the predecessor "Strong A" component is successively added to the synthesized replacement component obsolescence date until the resulting date is later than the constraint end date. This scenario can occur when the time between production events is larger in comparison to the "Strong A" component's procurement lifetime (L) or when the "waiting time" (Y_{R_k}) is larger than L. A production event must follow the obsolescence date for the synthesized replacement component otherwise no constraint is required. The scheme used to generate constraints for the synthesized replacement component that create a "Strong A" obsolescence event is graphically represented in Figure 3-9.

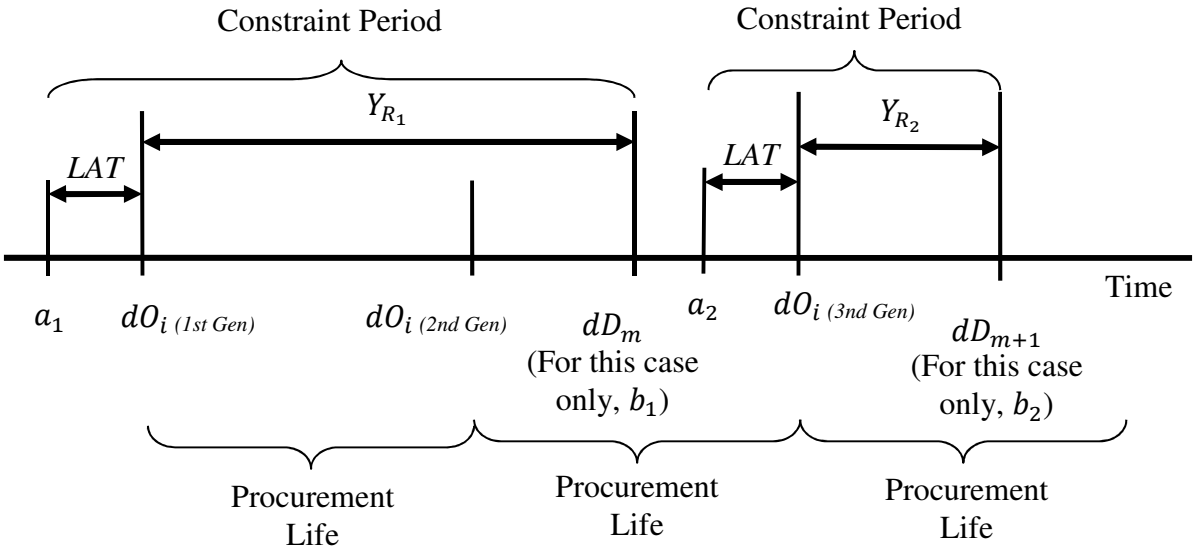


Figure 3-9: Graphical representation for generating constraints for synthesize replacement components that create "Strong A" obsolescence events.

In the case of a Strong B constraint, once an obsolescence date for the synthesized replacement component has been determined, steps 1 and 2 in the constraint generating algorithm are used to create the corresponding constraint in addition to determining if another synthesized replacement component is needed. The scheme used to generate constraints for the synthesized replacement component that creates a "Strong B" obsolescence event is graphically represented in Figure 3-10.

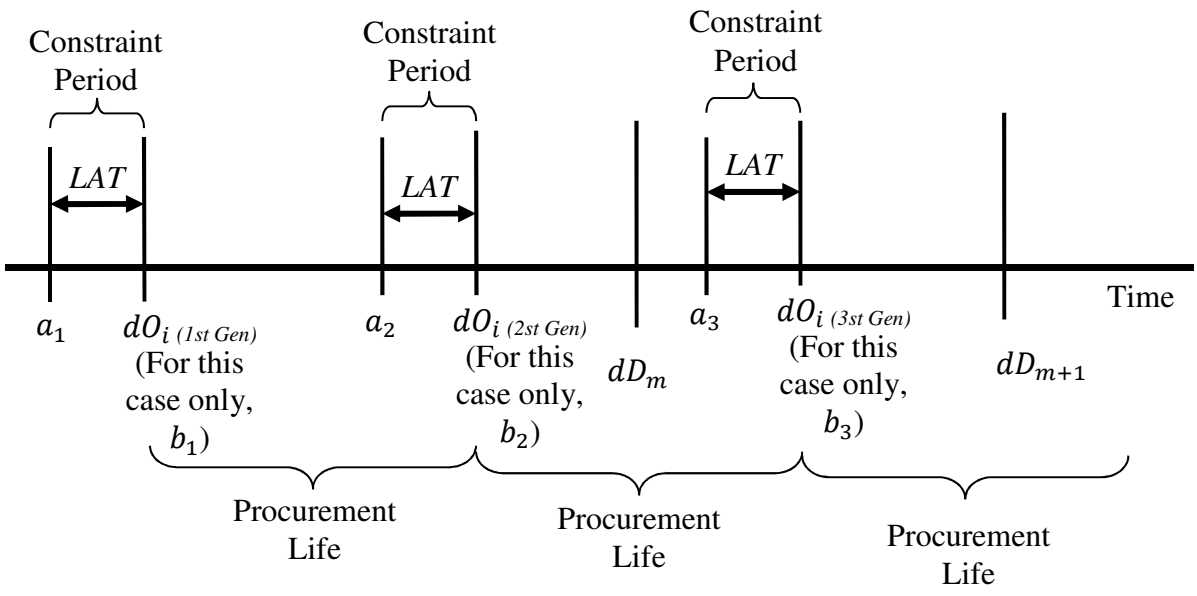


Figure 3-10: Graphical construction of constraints for synthesized replacement components that create "Strong B" obsolescence events.

For both examples shown in Figure 3-9 and Figure 3-10 the events, procurement life for the replacement component, and the "look ahead" time are all the same. The only thing different is the component's obsolescence event type. Looking at both examples it can be seen that constraints generated from "Strong A" obsolescence events have a longer constraint period; however, there are fewer constraints. "Strong B" obsolescence events have shorter constraint periods but have more constraints.

3.3.3 Step 3: Applying Constraints

In general, all constraints are applied to the design variable and joined with a logical "AND", i.e., all constraints must be satisfied for a design refresh plan to be considered viable; however, while all constraints must be satisfied, the way in which these temporal constraints are satisfied is different from what is commonly done in a conventional optimization problem. In a conventional optimization problem, there is an objective function expressed in terms of several independent variables. These variables are usually represented

as a vector and a unique set of values for each variable makes up a unique design. The constraints that are imposed on these design variables specifically limit the range of values each individual variable can take. This idea of a specific constraint limiting the value range of a specific variable (e.g., $g_1(x_1)$ is a specific constraint that limits the value range of specific variable x_1) is going to be known as the conventional way of applying constraints in this dissertation. As mentioned previously, these temporal constraints require a design refresh to complete within a bounded range of time; however, they do not specify any particular design refresh, thus any design refresh in the set of design refreshes that make up a design refresh plan can satisfy the constraint (since the content of the refresh in not predetermined, but rather is a function of when it occurs). For example, let the design refresh plan ($\bar{x}$) have exactly two design refreshes. Using Equation 1-1 as the objective function the optimization problem would be represented as:

$$\bar{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \ \bar{p} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ \vdots \end{bmatrix}$$

$$\min \quad f(\bar{x}, \bar{p}) = TC_M + TC_{DR} + TC_S$$

Keep in mind that $\bar{p}$ is just a vector of parameters that is here for thoroughness but not of interest in this discussion.

As an example of managing multiple constraints, let there be two temporal constraint periods, which cannot be converted into penalty functions, need to be satisfied. Some additional information is needed to determine the constraint start and end dates; however, for brevity let the constraints periods be defined as:

Constraint Period 1 = [$a_1; b_1$]

Constraint Period 2 = [$a_2; b_2$]

where the values a_k and b_k are related as:

$$a_k < b_k < a_{k+1} < b_{k+1} \quad for\ k = 1,2,\cdots,K$$

Using these values we can write constraints in a general form; however, neither constraint period 1 nor constraint period 2 identify a specific design refresh, we have to write all possible combinations of the constraints. With two variables and two constraints there are 4 possible ways these constraints can be applied.

$$\overrightarrow{g_1}(\bar{x}) = a_1 - x_1 \leq 0;\ \overleftarrow{g_1}(\bar{x}) = x_1 - b_1 \leq 0$$

$$\overrightarrow{g_1}(\bar{x}) = a_1 - x_2 \leq 0;\ \overleftarrow{g_1}(\bar{x}) = x_2 - b_1 \leq 0$$

$$\overrightarrow{g_2}(\bar{x}) = a_2 - x_1 \leq 0;\ \overleftarrow{g_2}(\bar{x}) = x_1 - b_2 \leq 0$$

$$\overrightarrow{g_2}(\bar{x}) = a_2 - x_2 \leq 0;\ \overleftarrow{g_2}(\bar{x}) = x_2 - b_2 \leq 0$$

Note: there is also a non-negative constraint imposed on the design variables and is represented as:

$$g(\bar{x}) - x_n \leq 0 \quad for\ n = 1,\cdots,N$$

This constraint will not be further included in the discussion since non-negative constraints are applied conventionally.

These constraints $\overrightarrow{g_1}(\bar{x})$; $\overleftarrow{g_1}(\bar{x})$ thru $\overrightarrow{g_2}(\bar{x})$; $\overleftarrow{g_2}(\bar{x})$ are plotted in Figure 3-11. A conventional application of these constraints is when the specific constraints limit the value range of a specific variable. It is apparent from Figure 3-11 that this problem is what is commonly known as an infeasible problem where there no region within the design space satisfies all the constraints (i.e., no overlap between all four regions).

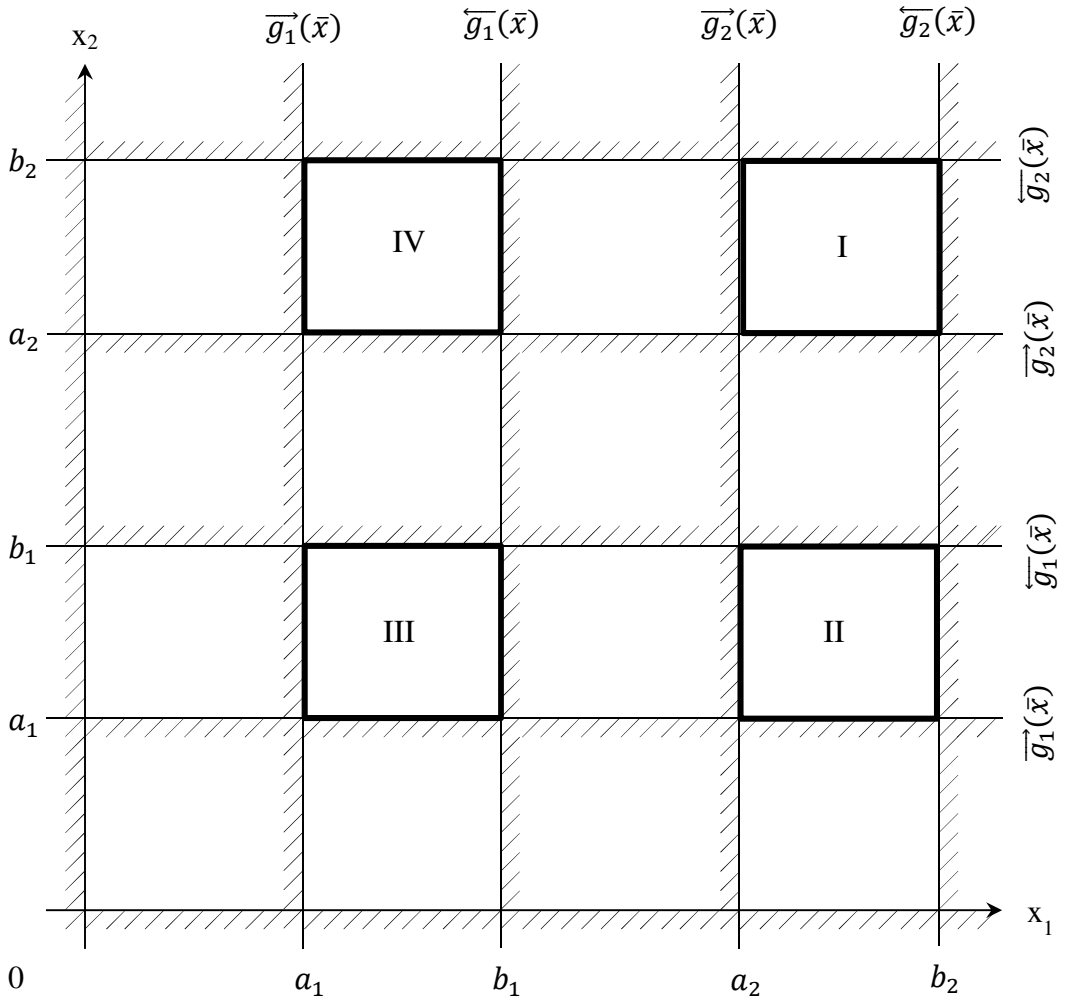


Figure 3-11: Constraints plotted and constraint overlap regions identified.

This is not an infeasible problem when dealing with temporal constraints described in this dissertation because of two reasons: 1) any of the design variables can satisfy any of the constraints; 2) the number of independent constraints is equal to the number of design variables. Reviewing Figure 3-11 will show four regions where a design refresh plan can exist and satisfy at least one constraint period and those regions are I and III. If a refresh design is within regions II or IV both constraint periods will be satisfied.

It is apparent that the conventional way of applying constraints is lacking a means of expressing this unconventional evaluation of constraints. Usually, if constraints are imposed on an optimization problem the following expression is included with the objective function:

$$\text{s.t.} \quad \overrightarrow{g_k}(\bar{x}) \leq 0 \qquad k = 1,\ldots,K$$
$$\overleftarrow{g_k}(\bar{x}) \leq 0$$

where K is number of constraints imposed on the design variables. This fails to express constraints that have the design variable specificity removed. One way to express this is by explicitly writing out all possible combinations that satisfy all the constraint periods and then joining each combinatorial set with a logical "OR" operator shown in Equation 3-9 as $\vee$. For example, in the example illustrated in Figure 3-11 there are two constraint periods that with respect to each design refresh variable are separated (i.e., since $a_k < b_k < a_{k+1} < b_{k+1} \quad for\, k = 1,2,\cdots,K$, there is no overlap between b_1 and a_2) and the two design refresh variables, so then the number of possible combinations is:

$$\frac{N!}{(K-N)!} \qquad \textbf{Equation 3-8}$$

where K is the number of constraints and N is the number of design refreshes. This of course only works if $N \geq K$ since multiple refreshes can reside in one constraint period but one refresh cannot reside in separated constraint periods (also the factorial of a negative number is undefined). Equation 3-9 shows the representation of one constraint period using the logical "OR" operator ($\vee$) to remove the design specificity.

Equation 3-9

$$\begin{bmatrix} \overrightarrow{\boldsymbol{g_1}}(\bar{\boldsymbol{x}}) = \boldsymbol{a_1} - \boldsymbol{x_1} \leq \boldsymbol{0}; & \overleftarrow{\boldsymbol{g_1}}(\bar{\boldsymbol{x}}) = \boldsymbol{x_1} - \boldsymbol{b_1} \leq \boldsymbol{0} \\ \overrightarrow{\boldsymbol{g_2}}(\bar{\boldsymbol{x}}) = \boldsymbol{a_2} - \boldsymbol{x_2} \leq \boldsymbol{0}; & \overleftarrow{\boldsymbol{g_2}}(\bar{\boldsymbol{x}}) = \boldsymbol{x_2} - \boldsymbol{b_2} \leq \boldsymbol{0} \end{bmatrix} \vee \begin{bmatrix} \overrightarrow{\boldsymbol{g_1}}(\bar{\boldsymbol{x}}) = \boldsymbol{a_1} - \boldsymbol{x_2} \leq \boldsymbol{0}; & \overleftarrow{\boldsymbol{g_1}}(\bar{\boldsymbol{x}}) = \boldsymbol{x_2} - \boldsymbol{b_1} \leq \boldsymbol{0} \\ \overrightarrow{\boldsymbol{g_2}}(\bar{\boldsymbol{x}}) = \boldsymbol{a_2} - \boldsymbol{x_1} \leq \boldsymbol{0}; & \overleftarrow{\boldsymbol{g_2}}(\bar{\boldsymbol{x}}) = \boldsymbol{x_1} - \boldsymbol{b_2} \leq \boldsymbol{0} \end{bmatrix}$$

In the case where $N < K$ a closer examination of the constraints would have to be made to determine if there are overlaps between constraints where one design refresh variable can satisfy multiple constraints, but of course the number of overlaps plus the number of separated constraints must be equal to or less than the number of design refresh variables. For example, let there be three constraint periods and only two designs refreshes; however, there is an overlap between constraint periods 2 and 3, so the relationship between the constraint periods is represented in the following way:

$$\text{Constraint Period 1} = [\, a_1;\, b_1]$$

$$\text{Constraint Period 2} = [\, a_2;\, b_2\,]$$

$$\text{Constraint Period 3} = [\, a_3;\, b_3\,]$$

where the values a_1, b_1, a_2, b_2, a_3 and b_3 are related as:

$$a_1 < b_1 < a_2 < a_3 < b_2 < b_3$$

Because constraint periods 2 and 3 share a common bounded range of values, they can share a common design variable. The constraint set can now be written as:

Equation 3-10

$$\begin{bmatrix} \overrightarrow{g_1}(\bar{x}) = a_1 - x_1 \le 0; \;\; \overleftarrow{g_1}(\bar{x}) = x_1 - b_1 \le 0 \\ \overrightarrow{g_3}(\bar{x}) = a_3 - x_2 \le 0; \;\; \overleftarrow{g_2}(\bar{x}) = x_2 - b_2 \le 0 \end{bmatrix} \vee \begin{bmatrix} \overrightarrow{g_1}(\bar{x}) = a_1 - x_2 \le 0; \;\; \overleftarrow{g_1}(\bar{x}) = x_2 - b_1 \le 0 \\ \overrightarrow{g_3}(\bar{x}) = a_3 - x_1 \le 0; \;\; \overleftarrow{g_2}(\bar{x}) = x_1 - b_2 \le 0 \end{bmatrix}$$

In the case where $N < K$ and the number of overlaps plus the number of separated constraints is still greater than K then the problem is truly over constrained and no design refresh plan with exactly two design refreshes will satisfy all constraints; however, this is under the assumption that there is no input uncertainty and that all dates, quantities, and costs are exactly correct. In reality nothing is ever really without uncertainty. Design refresh plans that

fail to satisfy all constraints without the presence of uncertainty can still be considered when input uncertainty is present.

The constraint implementation method shown as Equation 3-10 was presented in a constructive manner as a means of communicating the constraint application needs of the problem; however, a more formalized means of achieving this kind of constraint application can be found in non-linear generalized disjunctive programming (GDP), where the logical 'OR' operator is modeled using Boolean variables. These Boolean variables when paired with the terms of a constraint can ignore sets of constraints and perform the same effect as a logical 'OR' operator [56].

3.4 Constraint Implementation

The temporal constraint algorithm was implemented in a design refresh planning (DRP) tool called MOCA (Mitigation of Obsolescence Cost Analysis) [25], which is also used as the DRP test environment to study the effects of introducing constraints. MOCA is an event-based, stochastic, discrete event simulator, which means it creates a timeline, places events on the timeline, and accumulates costs as each event occurs. MOCA uses a brute force method for finding the best design refresh plan within the design space. It does this by first creating unique combinations of design refreshes, each one to complete its activities before the production of a new system begins; this particular methodology for generating design refreshes is known as a "just in time" design refresh approach. If there were m system production/manufacturing events, then MOCA would generate 2^m design refresh plans, each

of which is a unique combination of zero or more design refreshes; these are referred to as "candidate" design refresh plans.[12]

To implement the constraints developed in this dissertation into MOCA, the question of when should the constraints be applied in the life-cycle cost minimization process needs to be addressed. Constraints can be applied in three general ways. The first is a pre-processing operation where the constraints are imposed on the design point before the design point is evaluated by the objective function, which is advantageous since verifying a design point satisfies all constraints before the evaluation of the objective function saves on computational expense. The second is a concurrent operation where the constraints and objective function are verified and evaluated, respectively at the same time. The third is a post-processing operation where the design point is evaluated first by the objective function and then later verified with the constraints. In optimization theory, it is common for constraints to be converted into penalty functions and combined with the cost function so that both constraint violation and the cost function are minimized concurrently such as in the Lagrange Multiplier theorem and in Genetic Algorithms [43]. While the possibility to convert temporal constraints into penalty functions exists in DRP, not all temporal constraints can be converted into penalty functions because for some constraints no amount of violation is tolerated. Since the conversion of constraints into penalty functions is not always possible, and since a fair comparison between the design refresh plans is not achievable if done solely based on the severity of constraint violation, constraints are applied after the evaluation of all candidate design refresh plans by the objective function in what can be viewed as a post-processing operation.

[12] MOCA actually generates all 2^n design refresh plans, where n is the number of manufacturing events; however, it can optionally limit the generation to design refresh plans with a maximum number of design refreshes or with an exact number of design refreshes.

Chapter: 4 Modeling Constraint and Solution Uncertainty

As described in Chapter 3, temporal constraints are dependent on the obsolescence date of components and production dates of the system, which up until this section of the dissertation have been assumed to be exact; however, in reality there can be very large uncertainties associated with input data especially obsolescence dates. While including the uncertainty inherent in these dates makes for a model that better captures the behavior of design refresh management there is a concern that the variations in the obsolescence and production dates and consequently the variation in the constraint periods results in cases where none of the practical refresh plans[13] satisfying all constraints, all of the time; however, a design refresh plan that can satisfy all constraints, all the time. While theoretically, a refresh plan can always be found that satisfies all constraints all the time in the presence of uncertainties, in the real world there are several practical limitations on refresh plans and it is common for real systems that the actual set of candidate design refresh plans you have to select from will not include any plans that satisfy all the constraints all the time. This happens due to the fact that the plans you have to select from either: 1) conform to a particular management tradition, style, or culture (e.g., the use of a fixed frequency refresh planning scheme. See Section 4.3); and/or 2) they were generated by an algorithm that conforms to a set of generation rules (e.g., some design refresh planning methodologies use a "just in time" refresh approach) [25]. Therefore, it is valuable to be able to choose the best refresh plan from a set of refresh plans where none of the refresh plans satisfy all the criteria all the time.

[13] A refresh plan is a set of one or more refreshes where each refresh is at a unique date (more specifically, it has a unique completion date).

4.1 Modeling Uncertainty in Design Refresh Planning

DRP models that incorporate input uncertainty have been developed: albeit, their breadth of use are limited. Singh and Sandborn (2006) developed a DRP model and tool called the Mitigation of Obsolescence Cost Analysis (MOCA) which is a discrete event simulator that models input uncertainty using the Monte Carlo method [25]. Currently, it does not incorporate constraints that restrict the DRP activities temporally. The DRP model developed by Kumar and Saranga (2008) includes input uncertainty by utilizing a restless bandit model that is equivalent to a Markov decision process; however, it makes the assumption that once a component is redesigned (i.e., design refresh) it can be procured indefinitely [27]. This assumption confines the model's applicability to a limited set of problems.

These DRP models lack: constraints that restrict the DRP activities temporally; a means for modeling constraints whose bounds have uncertainty; a way of estimating the cost of violating constraints, and a method for deriving a probability of a design refresh plan failing to satisfy all constraints (i.e., a probability of failure).

4.1.1 Constraint Uncertainty

Uncertainty in constraints, specifically temporal constraints, have been modeled in various ways such as in chance-constrained programming [57] [58] and in simple temporal problems with preferences and probabilities (STP^3) [59]. These methods essentially prescribe a confidence level at or above which the probability that the design variable will satisfy the constraint must meet; however, knowing the appropriate confidence level to set is not straightforward and ultimately may lead to no design points meeting the confidence level

constraint. A more preferable arrangement is to have the constraint bounds' uncertainty modeled and making the probability of the design point satisfying the constraints as an objective.

When formulating constraints, uncertainty of the constraint bounds may not be an obvious concern, since for many constraints the bounds do not have uncertainty. For example, the "non-negativity" constraint has no uncertainty since the boundary between positive and negative (i.e., the position of zero) does not change under any circumstance. Consider the scenario of designing a carry-on luggage for air travel throughout various countries in the world. Different airlines use different planes of which each has a different maximum size for carry-on luggage. Let the maximum allowable thickness of the luggage be consistent across all planes; however, the width and height of the luggage can very. Figure 4-1 is a plot of the design space where $x_1$is the carry-on width, x_2 is the carry-on height, *a* is the average maximum carry-on width across all planes, and *b* is the average maximum carry-on height across all planes. The objective is to maximize the carry-on width and height surface area while staying below the maximum area constraint threshold of *ab*; however, there is another concern which is that the maximum area constraint has uncertainty. It is preferable to be 90% confident that the carry-on luggage will fit in all planes. In addition, since you are designing the carry-on luggage, anything manufactured will have a degree of uncertainty, so the width and height of the luggage also has uncertainty. The question becomes how to maximize carry-on width-height area while also maximizing confidence that the luggage will fit into all planes? This example illustrates the same idea of selecting a design refresh plan that will minimize life-cycle cost while also minimizing the plan's probability of failure in satisfying all constraints. The example had only one constraint;

however, the DRP problem is capable of having multiple constraints. How can the probability of failure be determined when multiple constraints with uncertainty are imposed?

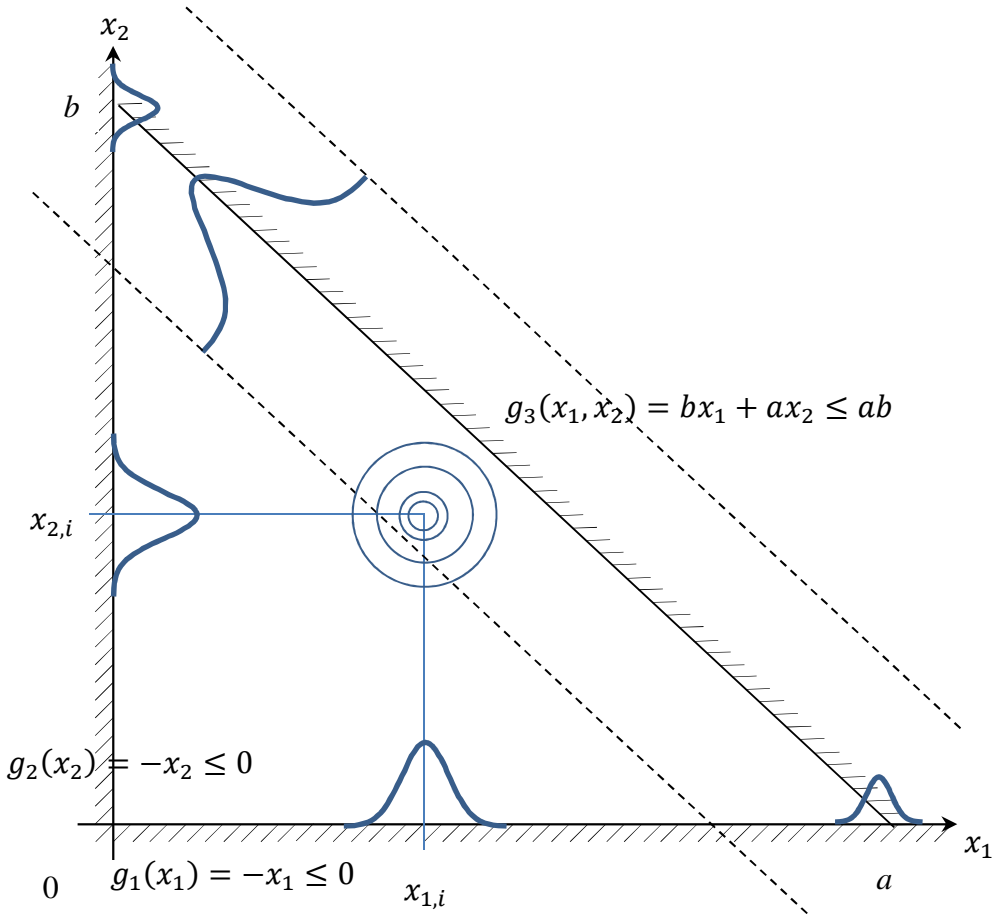


Figure 4-1: Design space for carry-on luggage width and height.

One way to determine the probability of a design refresh plan failing to satisfy a set of imposed constraints is to calculate the probabilities of individual constraints failing to be satisfied by the design refresh plan and then computing the product of those failure probabilities. Before probabilities can be calculated, certain assumptions about the parameters and variables used in the modeling of the events have to be declared. Table 4-1 goes over the various assumptions on the DRP model parameters and variable.

Table 4-1: Parameter/Variable assumptions

Parameter / Variable	Assumptions
Date of Component Obsolescence, dO_i [Parameter]	This parameter is independently random. The reasoning behind this is that the manufacturer has control over when to continue or stop producing the component; however, other outside influences not in control by the manufacturer will affect when the component is actually obsoleted.
Date of Design Refresh, x_n [Variable]	This variable is independently random with respect to the other parameters. While a design refresh completion date may be selected, the actual completion date is not in control by the system manager because the uncertainty of new technology and the time it takes to design refresh for it is uncertain. It is also assumed that a design refresh will take place regardless if it does or does not resolve the obsolescence issue of the nearest component obsolescence event.
Date of Production Start, dD_m [Parameter]	This parameter is independently random with respect to the other variable/parameters. While a production start date can be scheduled, it will be assumed that future influences such as natural disasters, economic uncertainties and changing labor force overtime result in an uncertain production start date. It is also assumed that production will start regardless if a design refresh fails to complete before the production start date.

With the assumptions in Table 4-1 established, consider the special case where the time a design refresh "looks-ahead" into the future at the completion of the design refresh event to proactively resolve forecasted obsolescence is set to zero, meaning at the beginning of a design refresh, only components that are already obsolete are resolved and no components that are forecasted to be obsolete in the future (within the "look-ahead" time) are resolved. In this special case, a "Strong A" constraint that requires a design refresh to resolve

the obsolescence of a component and complete its activities before the next production event is imposed. If however, the production, constraint, and design refresh dates are normally distributed, they would appear as shown in Figure 4-2, given the component obsolescence date has a mean of μ_{D_O} and a standard deviation of σ_{D_O}; the design refresh date has a mean date of μ_x and a standard deviation of σ_x; finally the system production date has a mean of μ_{D_P} and a standard deviation of σ_{D_P}.

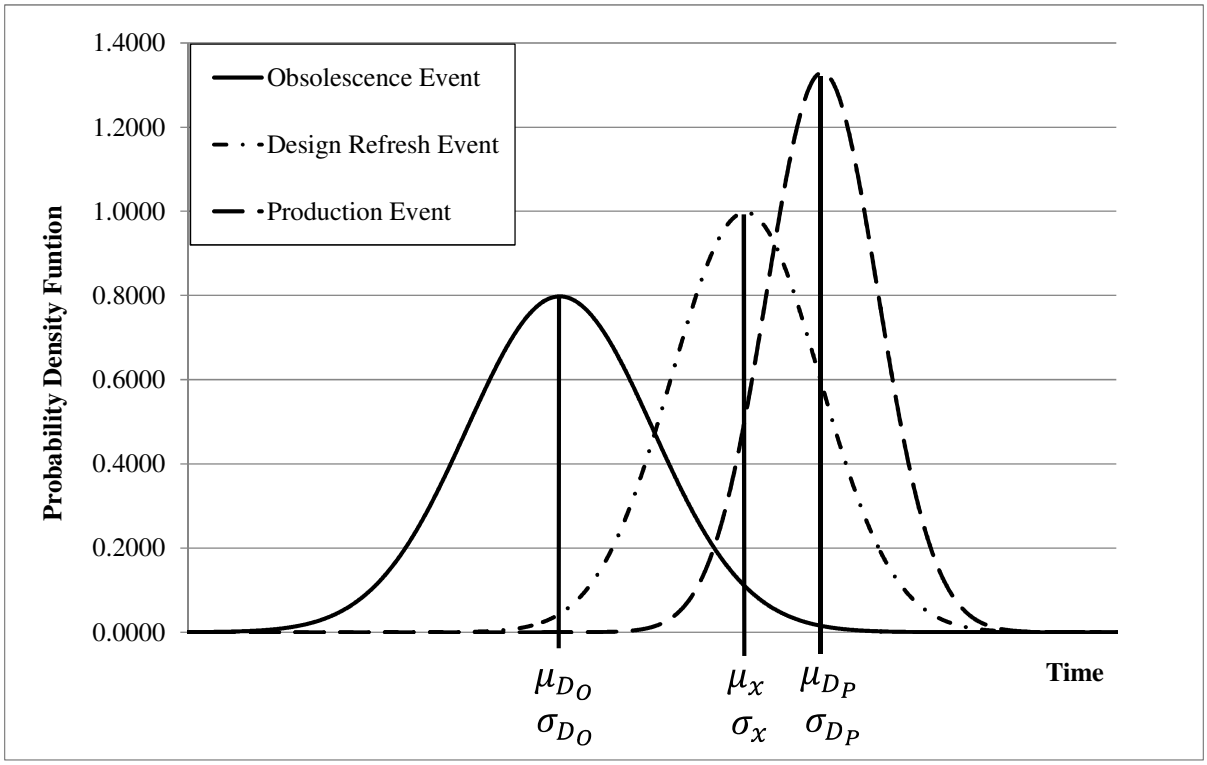


Figure 4-2: Illustration of refresh date and constraint date distribution overlap.

Figure 4-2 illustrates this example and shows the distribution of dates for the system production event, a component obsolescence event, and a planned design refresh event.

To determine the probability that a design refresh will fail to satisfy the imposed constraint, which requires it to complete its activities between the obsolescence event of the

component and the succeeding production event is done by calculating the probability that the design refresh will satisfy the constraint, which will be called "design refresh reliability" and then finding the difference from unity to get the probability of failure. To calculate the design refresh reliability ($\breve{R}$), a probability must be formulated relating to the order of events that satisfy the constraint. For example, the date a component goes obsolete (dO_i) must occur before a design refresh (x_n) finishes its activities so that the design refresh can resolve the problem created by the obsolescence event of the component. A design refresh satisfies the constraint when the design refresh completes after the component obsolescence event occurs. This design refresh reliability is represented as the distribution area overlap shown on Figure 4-3; however, the probability that $dO_i < x_n$ is not the actual overlap area.

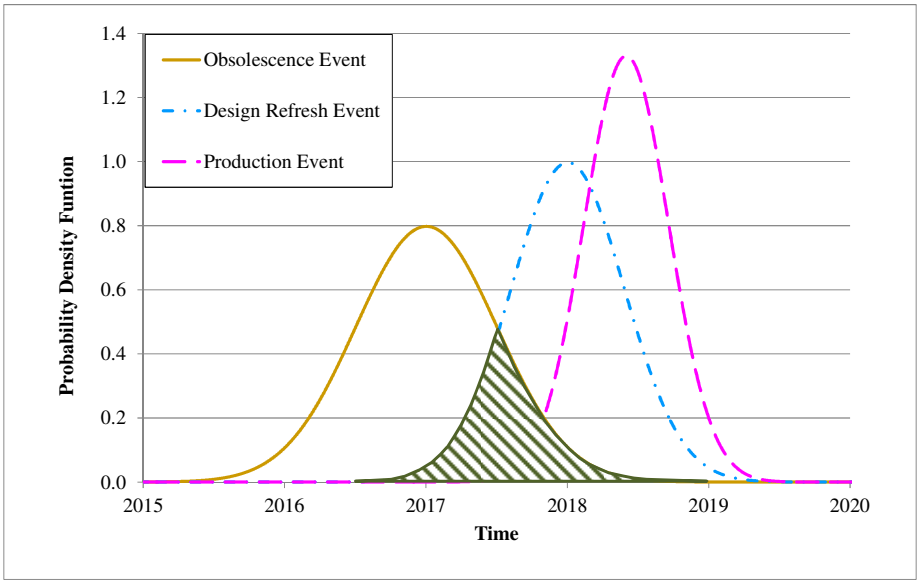


Figure 4-3: The overlap area between the obsolescence date distribution and the design refresh date distribution.

The second condition that is required for the design refresh to satisfy the constraint is that it should complete its activities before the date of production (dD_m) for a new system instance begins. The design refresh satisfies this constraint when the design refresh completes its activities after the production of a new system begins and this is represented as

the overlap area of the two distributions shown in Figure 4-4, but again this probability is not actually the overlapped area.

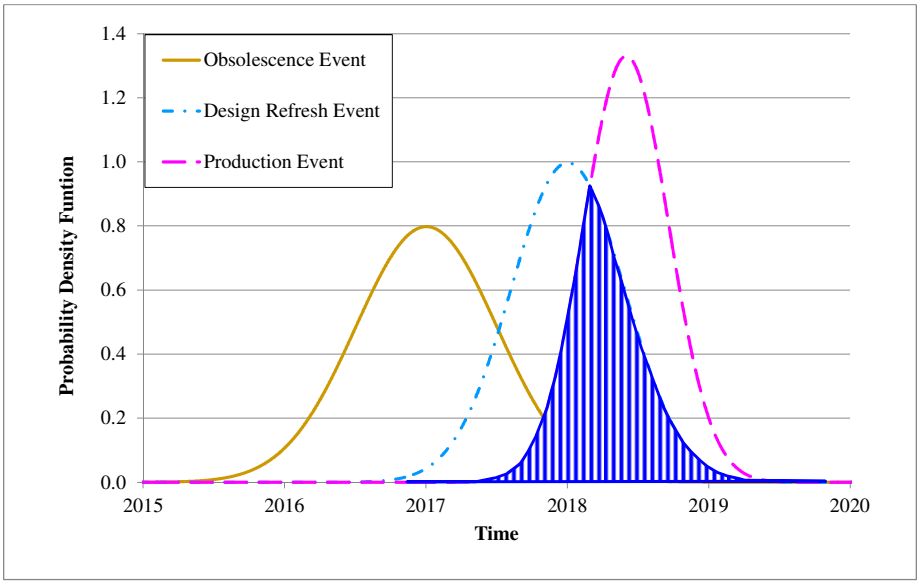


Figure 4-4: The overlap area between the design refresh and production date distributions.

Finally, the last condition required for the design refresh to satisfy the constraint is that the component obsolescence date (dO_i) should take place before the production of a new system (dD_m) this probability of occurring is represented as the overlapped area of the two distributions shown in Figure 4-5.

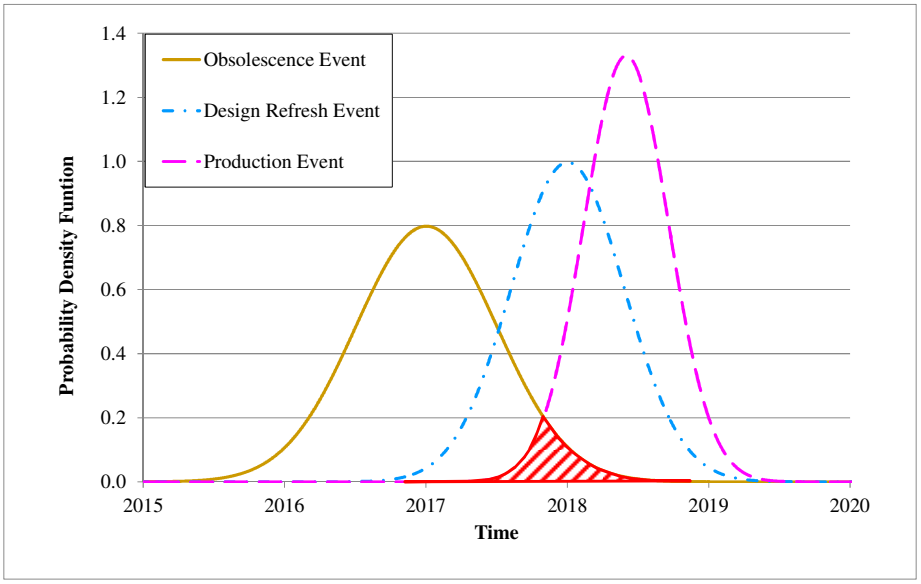


Figure 4-5: The overlap area between the obsolescence and production date distributions.

The final probability that a design refresh date will satisfy the condition that it come after the obsolescence date of the component and before the start of production is given in Equation 4-1.

$$\breve{R}_k = P(dO_i < x_n < dD_m) \quad \textbf{Equation 4-1}$$

$$a_k < x_n < b_k$$

$$p_f = 1 - \breve{R} \quad \textbf{Equation 4-2}$$

gives the probability (p_f) that a design refresh (x_n) fails to satisfy the constraint period. For multiple constraints, each constraint has a design refresh reliability, $\breve{R}_k$ where the k denotes the constraint period within a set of K number of constraint periods. Computing the product of all design refresh reliabilities equals the reliability for the entire set of constraints ($\breve{\breve{R}}$):

$$\breve{\breve{R}} = \prod_{k=1}^{K} \breve{R}_k = \breve{R}_1 \breve{R}_2 \cdots \breve{R}_K \quad \textbf{Equation 4-3}$$

The result of Equation 4-3 is use to find the probability of failure for the design refresh plan.

$$P_f = 1 - \breve{\breve{R}} \quad \textbf{Equation 4-4}$$

A less efficient but more general method for approximating the design refresh plan failure probability is to use a Monte Carlo method and run the DRP model a statistically significant number of times (N_{mc})[14], and determine the fraction of runs in which the refresh plan failed (N_f) to satisfy all constraints (see Equation 4-5). Note, the sampled data input into the life-cycle cost function is the same sampled data used to generate all constraints for each

[14] See Appendix C for more information on statistical significance

Monte Carlo simulation, thus the constraints are rebuilt using the sampled dates for the component obsolescence, design refresh, and production events (See Appendix B).

$$P_f = \frac{N_f}{N_{mc}}$$ **Equation 4-5**

The distributions in this section are normally distributed for illustration purposes; however, the ideas presented here do not depend on the type of distribution used to model the uncertainty of input data.

4.2 Best Design Refresh Plan

Defining what the best design refresh plan means out of a set of candidate design refresh plans is straight forward for unconstrained problems – the best design refresh plan is the one that has the lowest associated cost. Applying constraints to the problem makes finding the best design refresh plan more challenging; however, a constrained problem without input uncertainty present is still straightforward since the best design refresh plan is the one that has the lowest associated cost that satisfies all constraints. When the problem is constrained in the presence of input uncertainty, the task of finding the "best" design refresh plan becomes complicated since it is not clear as to what is meant by the word "best." In other words, when input uncertainty is present, design refresh plans can potentially have a probability that they will not satisfy all constraints so the meaning of "best" can apply to a design refresh plan that has the lowest associated life-cycle cost but also it applies to the design refresh plan that has the lowest probability of failing to satisfy all constraints.

If the probability of the design refresh plan failing to satisfy all constraints can be viewed as a second objective, then both life-cycle cost and probability of failure can be plotted together to obtain a criterion space. Figure 4-6 is a MOCA plot of life-cycle cost

versus probability of failure. The example data used here is taken from an actual DRP analysis for real system. Figure 4-6 demonstrates the possible case where the penalty cost for violating a constraint is lower in comparison to the cost of performing design refreshes. A design refresh plan with more design refreshes is more likely to satisfy more constraints, which is reflected in the design refresh plan's lower probability of failure; however, with the high cost of performing a design refresh its life-cycle cost is high. A design refresh plan with a low number of design refreshes in comparison will have less design refreshes to satisfy all the constraints, and thus has a higher probability of failure; however, since violating constraints is cheap (i.e., penalty cost is low compared to the cost of design refresh) it results in a lower life-cycle cost.

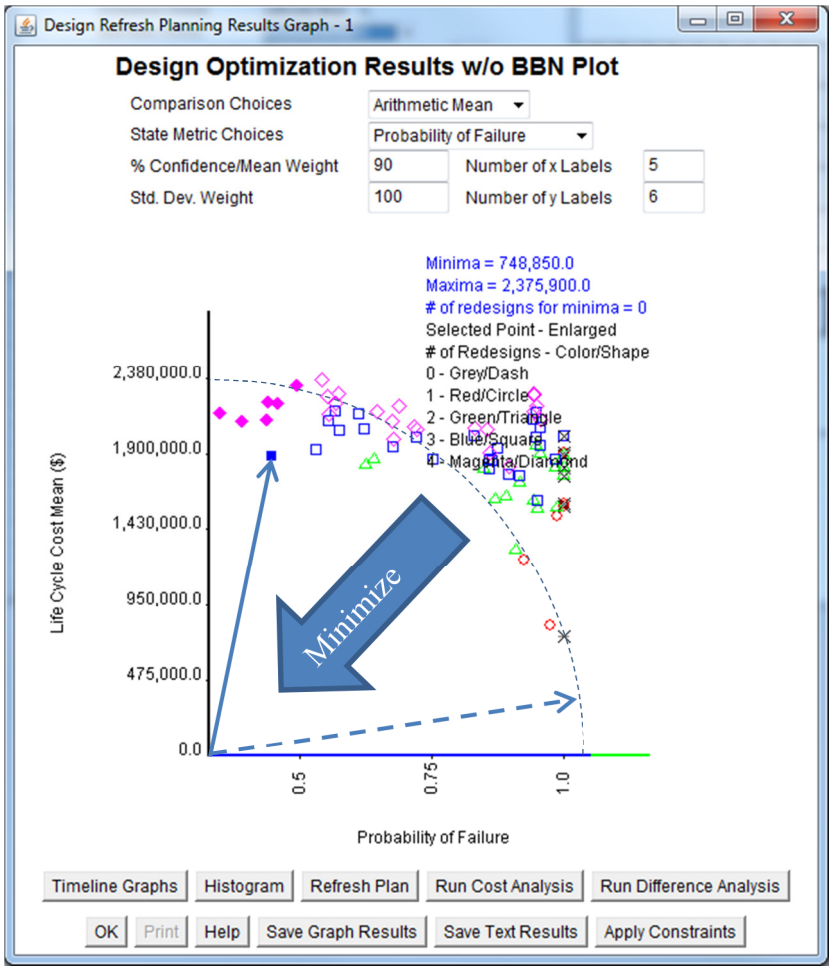


Figure 4-6: Graphical representation of life-cycle and design refresh plan probability of failure minimization problem in the criterion space.[15] Penalty costs are less compared to design refresh costs. The life-cycle cost and probability of failure have not been normalized.

The example case shown in Figure 4-6 is one with a Pareto frontier, which can be handled in a number ways such as a weighted sum method or utility function method [60]; however, if any plan on the Pareto frontier is acceptable, then wouldn't the "best" design refresh plan still be the one with the lowest life-cycle cost? If nothing else is affected by the violation of constraints then the answer would be yes, but violating a constraint (one that can be violated, i.e., a penalty function) does not just result in a penalty cost in practice. For

[15] While this is an actual results output graph from MOCA (less the arrows and drawn lines), the vertical-horizontal axis intersection has been modified to emphasize the idea of a Pareto optimality, which is defined in this dissertation as a design refresh plan $\bar{x}$ is called Pareto optimal if there is no other point $\bar{x}$ in the set of candidate design refresh plans that reduces at least one objective without increasing another one.

example, if a credit card company gave a period of time to pay off a balance, and you didn't make the payment within the constraint time period, then a late payment fee would not be the only thing you would incur – in addition, you might also receive a higher annual payment rate and/or a delinquency mark placed on your credit report, which can result in a decrease of your credit score and potentially prevent you from getting a better interest rate in the future.

When the example analysis is performed again but with a penalty cost that is significantly higher than the cost of performing a design refresh, the effect is reversed and the results essentially flip vertically, which is presented in Figure 4-7. In this case, the "best" design refresh plan is obvious since the lowest life-cycle cost design refresh plan also has the lowest probability of failure.

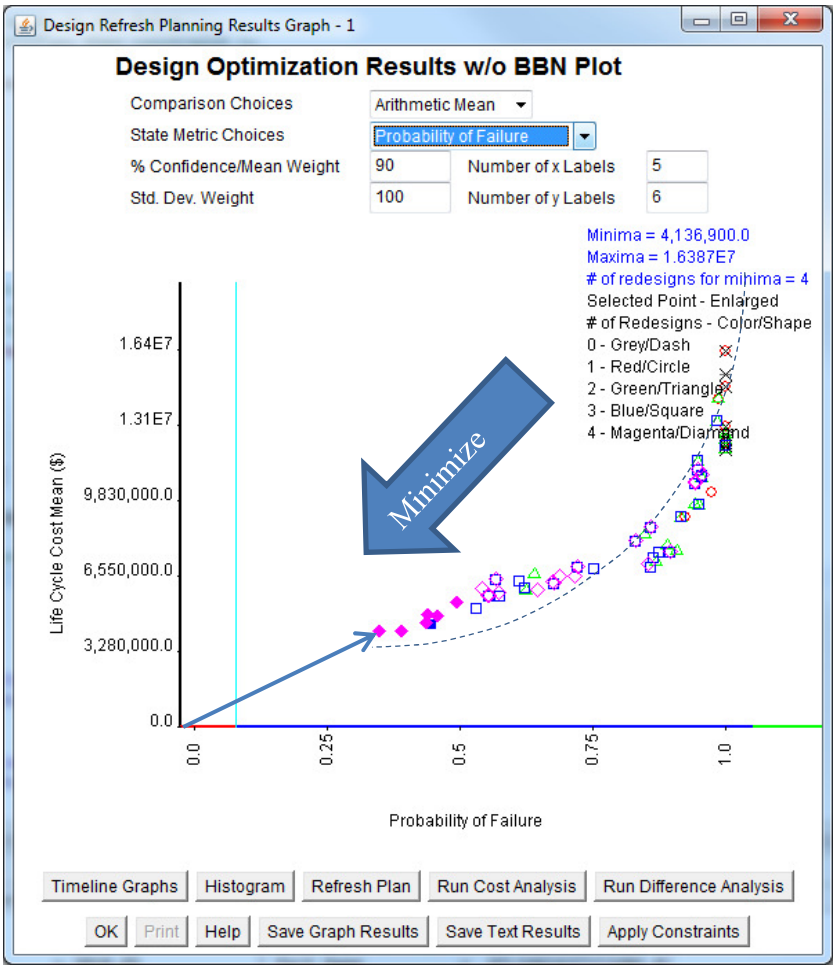


Figure 4-7: Graphical representation of life-cycle and design refresh plan probability of failure minimization problem in the criterion space. Penalty costs are greater compared to design refresh costs. The life-cycle cost and probability of failure have not been normalized.

4.3 Deriving the Cost for a Violated Constraint

When a design violates a constraint the design is removed from the feasible set of designs; however, there is the possibility that no design satisfies all constraints resulting in no feasible designs. If a single design must be selected, how can the best design be selected out of a set of infeasible designs? One way would be to evaluate each design using the objective/cost function and rank each design based on its cost value. In the case with design refresh planning, each design is a design refresh plan and the cost value of each design refresh plan is its associated life-cycle cost. In order to correctly cost a design refresh plan

the costs associated with violating a constraint must be included. For example, if an imposed constraint states that a design refresh event must always complete its activities before the start of a production event (equivalently, a production event must always start after a design refresh's activities end) then what costs should be associated to the event where a design refresh event completes its activities after a production event? It is these constraint violation associated costs that are needed in order to cost all design refreshes fairly. Accuracy (absolute value) in costing these constraint violation associated costs is desired; however, it is not necessary since there are many costs shared by both the constraint satisfied and violated scenarios which will cancel out (i.e., "wash" out) when a difference analysis is performed.

4.3.1 Implementation of Constraint Violation Exterior Penalty Cost[16]

The actual cost of violating a constraint is complex, since depending on the constraint; several costs due to the violation can come from various sources and in different forms. For example, consider again the problem where a credit lender sets a period of time to pay a balance on the loan amount. Let the objective function be the amount of money in a high interest savings account and the design variable is the time at which the debt repayment is made. The goal is to maximize the money within the tiered high interest savings account where the interest increases with different levels of saved money. Holding the highest amount of money in the savings account will maximize the objective function. Furthermore, the credit lender does not want early payment since it would have to pay additional tax on the average daily balance of an escrow account, so the credit lender will penalize any early

[16] Exterior Penalty: A penalty function in which the associated algorithm generates infeasible points, approaching feasibility in the limit towards a solution [61].

payment. Late payments are penalized more severely than late payments. Consider the case where the payment does not satisfy the constraint, and a late fee (i.e., penalty cost) is incurred. Just considering the problem, this is the only penalty cost; however, in actuality other important consequences have occurred as a result of this constraint violation that need to be considered such as the credit lender increases the borrowing annual percentage rate, and/or the credit reporting agencies are notified by the late payment and decrease the borrower's credit score affecting other borrowing rates. This example problem can be extended to a payment for services rendered scenario. Let the credit lender in this example problem be an electric utility provider who also issues early and late payment penalties. If the payment constraint is violated not only would a fee be imposed, but also the power is shut off, which for a business results in lost revenue, payment of wages to salary workers but no work is being done, reputation damage, and other impacts. The penalty cost example given in the examples can be of two forms: fixed and variable. The fixed and variable penalty cost is issued every time the constraint is violated but variable cost is based on the distance from the constraint to the nearest design point. The magnitudes of the fixed and variable penalty costs are based on the constraint that was violated. To see the formulation of a comprehensive constraint violation (i.e., penalty) cost model see Section 6.3.3.

To implement this idea in for the model developed in this dissertation, a fixed and variable cost, which is proportional to the distance the constraint is from the nearest design refresh is added to the life-cycle cost when a constraint is violated. For this implementation, the penalty cost magnitudes are not changed with respect the obsolescence event type that made the constraint.

$$v = u + a\Delta t$$ **Equation 4-6**

where v is the total penalty cost per constraint violated, u is the fixed penalty cost per constraint violated, a is the coefficient for the variable penalty cost per unit time per constraint violated term, and Δt is the shortest time distance from the constraint to the nearest design refresh completion date. Figure 4-8 illustrations how the Δt is determined per constraint violated. Consider a design refresh plan $\bar{x} = [x_1, x_2, x_3]$ which does not satisfy the shown constraint period. To find the shortest time distance (Δt), all time distances are computed for all the design refreshes in the design refresh plan. Then Δt is the minimum of the time distances in the set (see Equation 4-7).

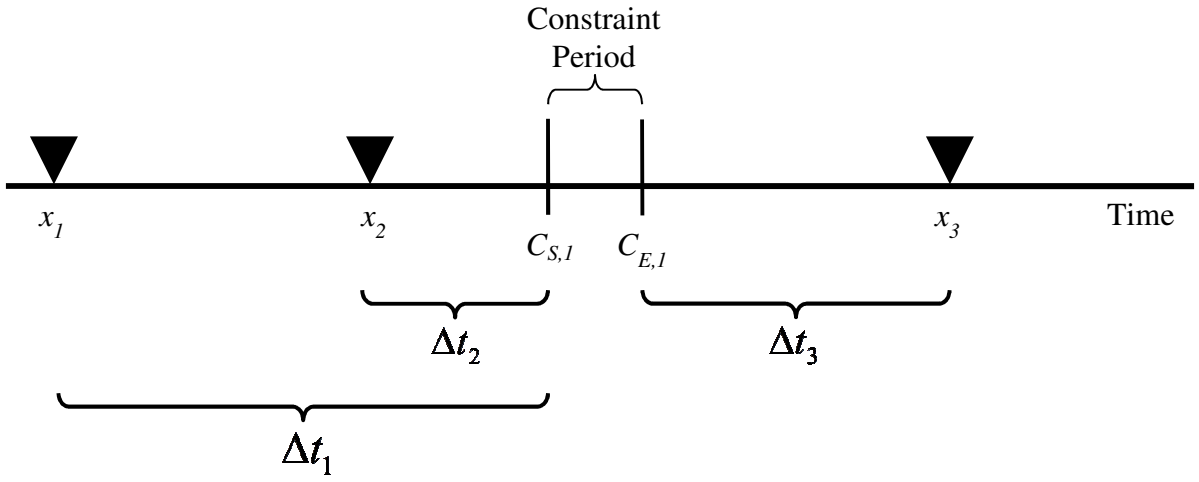


Figure 4-8: Graphical representation of how the shortest Δt is found.

$$\Delta t = \min(\Delta t_1, \Delta t_2, \cdots, \Delta t_r)$$ **Equation 4-7**

where *r* is the number design refreshes in a design refresh plan.

Chapter: 5 Temporal Constraint Application Case Studies

In order to use the constraint synthesis and management methodology described in this dissertation, it has been incorporated into the existing MOCA design refresh planning environment. Using the methodology developed in this dissertation, the constraint management capability in MOCA has been used in the evaluation of design refresh planning solutions for many different systems including: medical laboratory equipment (an In Vitro Diagnostic (IVD) company), ship-board communications systems (US Navy SPAWAR), tower communications systems (SBINET, Dept. of Homeland Security), V-22 avionics (US Navy), and others.

Since the input data for the real systems listed above is in general proprietary or otherwise controlled, this chapter provides a detailed discussion of one tutorial case study based on fictitious system data. This case demonstrates how to generate temporal constraints based on Weak, Strong A and Strong B obsolescence event types as well as how to apply them. In addition, the results from two real-world case studies that exemplify the use of Strong A and Strong B obsolescence event types are summarized.

5.1 Demonstration Case Study (All Obsolescence Event Types)

This case study is based on a portion of a fictitious communications system consisting of one server cabinet with several racks. The entire system is represented by a bill of materials with a total of 79 components. This communications system is sustainment-dominated and includes supporting as well as producing several instances of the server cabinet design. Table 5-1 provides information on the scheduled production of the

communications system. All production activities are planned to be completed in the month of January for each scheduled production year.

Table 5-1: The production dates and associated production quantities make up the planned production schedule.

Production Year	2007	2008	2009	2015
Production Quantity	4	4	4	2

In this case study, the system limitations consist of an information security policy and the Restriction of Hazardous Substances Directive (RoHS). The European legislation called the Restriction of Hazardous Substances (RoHS) Directive [62] regulates many of the commonly used substances in electronics and restricts the use of several materials deemed hazardous by the European Union (EU). The most problematic material for electronic systems is lead, which historically is a primary ingredient in solder. The legislation only pertains to electronic systems sold in the EU after July 1, 2006 so any system fielded prior to July 1, 2006 with non-compliant electronic components can continue operating and be maintained with non-compliant components; however, new instances of the system to be manufactured and sold in the EU market must comply with the RoHS directive by ensuring that every component and subsystem is RoHS compliant regardless of the availability of non-compliant components.

The objective of this study is to show the effect of applying explicit constraints to the DRP process on the selection of the best design refresh plan. In order to demonstrate the effect of applying constraints on a system being managed by the DRP process, a DRP modeling test-bed is required.

For this case study, the DRP modeling environment used is a DRP software tool called MOCA (Mitigation of Obsolescence Cost Analysis) [25], which is a DRP methodology for strategic management of systems affected by DMSMS. The MOCA model

utilizes input data in terms of hardware and software, and determines the life cycle code of multiple refreshes coupled with the reactive mitigation approaches. MOCA takes as its input the bill of materials (BOM) for a given system, along with the procurement cost and forecasted obsolescence dates or procurement lifetimes of the individual components.[17]

This case study has two implicit limitations that need to be transformed into explicit DRP constraints. The first implicit limitation for this case study is an information security policy that states "No software regardless of function is permitted to operate beyond its end of support life. Exemptions to this policy may be used beyond their end of support life; however, new systems may not be built with the exemptions. Exemptions include: drivers, firmware, and BIOS." This implicit limitation restricts any and all software used in the system. With the component types identified, the system that is composed of 79 total components is searched for the matching component types. There were 12 components identified with the component types restricted by the implicit limitation. Next obsolescence event definitions were assigned to the 12 components based on how their obsolescence events affected the system, according to this implicit limitation.

The second implicit limitation for this case study is the Restriction of Hazardous Substances Directive (RoHS) which states "These Regulations implement EU Directive 2002/95 that bans the placing on the EU market of new electrical and electronic equipment containing more than agreed levels of lead," and other hazardous materials. This implicit limitation also has exemptions as well as exclusions; however, none apply in this case study. This implicit limitation identifies any electronic component that is not RoHS compliant as the components being restricted from being built into new systems. Performing the same

[17] Several obsolescence forecasting methods can be used as input to MOCA [25], [18]- [24]. Obsolescence forecasts for electronic components are also available from several commercial sources.

method used previously on the first implicit limitation, the system's component list is reviewed and components that are restricted by the limitation are assigned obsolescence event definitions. There were 7 components identified and assigned obsolescence event definitions based on this implicit limitation.

For the sake of brevity, only 3 components will be used to demonstrate the formulation of explicit temporal constraints for the DRP process. Table 5-2 provides obsolescence information on the three example components.

Table 5-2: Subset of components whose constraint synthesis will be demonstrated in this case study. The components listed are 3 out of the 19 total identified components with component types restricted by the two implicit limitations.

Component Name	Component Type	Obsolescence Date[18]	Procurement Lifetime (years)	Obsolescence Event Type	RoHS Compliant
Fan Controller Driver	Driver	2007.5	10	Strong A	N/A
Fan Controller	Hardware	2018	30	Weak	No
Office 2000 Professional	Software	2009.54	10	Strong B	N/A

The assumptions for the DRP process are: look-ahead time (*LAT*) is set to 3 years, RoHS compliance date of January 1, 2014 (D_C = 2014.0), an end of support date (*EOS*) of 2020, an analysis period from 2005 to 2020, and a replacement component life cycle code of 2 (I_R = 2). It will be assumed for this system that there are no penalty costs or fees associated with violating a constraint. An example of these penalty costs is when a design refresh occurs after a production date causing a delay in production which results in a penalty cost.

The constraint generating algorithm is used to create the explicit constraints for each of the three components listed in Table 5-2. A detailed explanation of the formulation of the explicit constraints for each of the components is rather long and tedious for this dissertation; however, the resulting explicit constraints are shown in Table 5-3. When looking at the

[18] Date Representation – to simplify calculations all dates have been represented as the number year plus the fraction of the year that the date occurs. For example, the date July 16, 2007 is the represented as 2007.54 since July 16 is the 197th day out of the year so the year fraction is 197/365≈0.54.

resulting explicit constraints keep these three ideas in mind: 1) the Fan Controller component in Table 5-2 appears that it should not require a constraint because it is labeled as having a "Weak" obsolescence event type; however, since the Fan Controller is not RoHS compliant a constraint is required; 2) the constraint generating process does not stop with the original component, rather it stops with the replacement components that replace the original component; and 3) the "Strong B" obsolescence event types require backfits which need to be defined. Table 5-4 show the backfits created from the "Strong B" obsolescence event types.

Table 5-3: Constraints generated from the "Strong" components listed in 2 and their synthesized replacements.

Affected Component	Component Generation	Start Date (a_k) Bound (See Equation 5)	End Date (b_k) Bound (See Equation 7)
Fan Controller Driver	Original	$g_1(x)=x-2004.50\leq 0$	$g_2(x)=2008.00-x\leq 0$
Fan Controller	Original	$g_3(x)=x-2011.00\leq 0$	$g_4(x)=2015.00-x\leq 0$
Office 2000 Professional	Original	$g_5(x)=x-2006.54\leq 0$	$g_6(x)=2009.54-x\leq 0$
Office 2000 Professional	Replacement	$g_7(x)=x-2015.54\leq 0$	$g_8(x)=2017.54-x\leq 0$

Table 5-4: Production events used to implement backfits required for Strong B events.

Production Date (dD_m)	Production Quantity	Affecting Component	Component Generation
2009.54	12	Office 2000 Professional	Original
2017.54	14	Office 2000 Professional	Replacement

Figure 5-1 shows the results of the MOCA analysis of the example described in this section without applying constraints and without uncertainty[19]. Figure 5-2 shows the results of the MOCA analysis with the above generated constraints applied along with 5 other "Strong A" constraints. The horizontal axis of the graph shows the mean date for each refresh plan (each point is plotted at the mean of all the refresh dates in the plan) and the

[19] In order to produce a finite number of candidate design refresh dates, MOCA uses a "just in time" refresh policy in which the only allowed points in time where a refresh can finish are just before demand for systems, e.g., production events or spares demand. This assumption is discussed in detail in [25].

vertical axis shows the corresponding total life cycle cost. The data points each represent unique design refresh plans (unique combinations of design refreshes). The shape of the data point indicates how many design refreshes are in the refresh plan. The filled circle is a single refresh, the triangles have two refreshes in their plans, the square represents plans with three refreshes in them, etc. The rectangle (dash) is the zero refresh plan, which has zero refresh dates (i.e., all obsolescence is managed with lifetime buys for this example) and acts as a comparison life-cycle cost between doing nothing (i.e., zero refresh) and doing something (i.e., one or more refreshes).

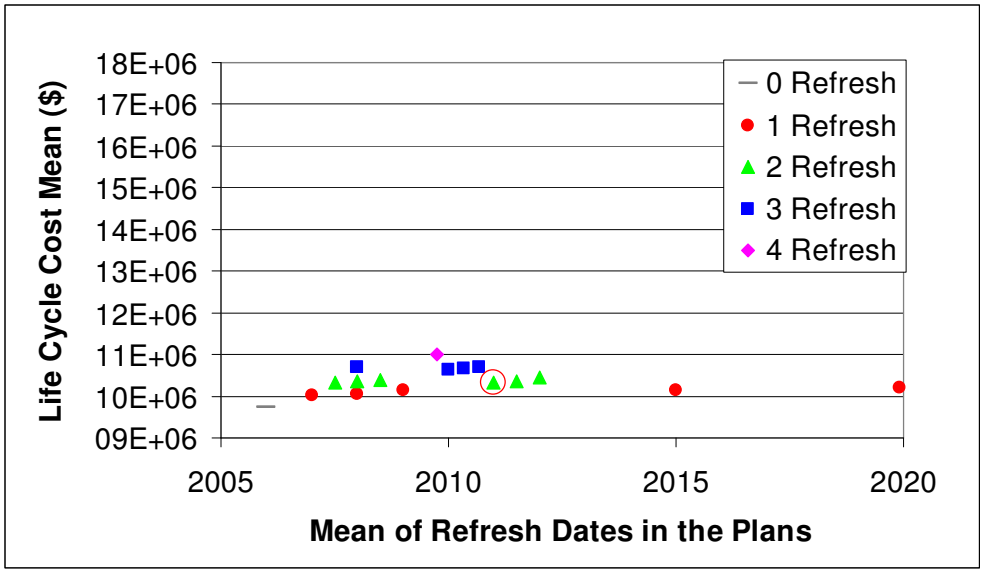


Figure 5-1: MOCA generated refresh plan mean dates versus life-cycle cost for the case study system with no constraints applied. The data point for each refresh plan is plotted at the mean of the refresh dates in the respective plans.

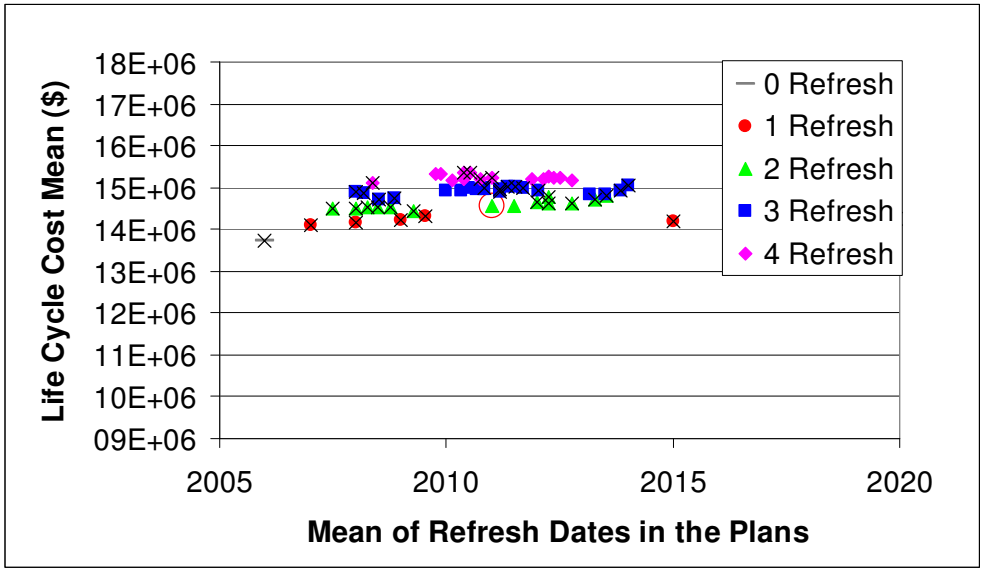


Figure 5-2: MOCA generated refresh plan mean dates versus life-cycle cost for the case study system with a deterministic application of the generated constraints. The least costly plan that satisfies all constraints is circled. Note, this figure is scaled the same as Figure 5-3.

Figure 5-1 and Figure 5-2 clearly demonstrate the effect of introducing constraints to the design refresh planning process. The increase in the number of refresh plans from 17 to 58 is due to the additional production events that were added to implement the "Strong B" backfits. Figure 5-2 shows that many refresh design plans can be created; however, once temporal constraints that reflect system limitations are applied only a few plans remain viable, i.e., satisfy all the constraints. In this case, 18 plans ranging from two to four refreshes per plan are viable out of a total of 58 plans. The violating plans are crossed out. The least expensive viable plan has two refreshes at 2007 and 2015 (triangle data point), which is circled in Figures 5-1 and 5-2. Note, the best plan without constraints applied is the zero refresh plan.

So far, the case study has assumed that there are no uncertainties associated with the data describing the system. The case study analysis is performed assuming that all dates in Table 5-2 are the mean of normal distributions (μ^*), all with a standard deviation of one year (σ^*). The uncertainty analysis method used does not require that the uncertainty inputs be

represented as normal distributions (or symmetric distribution) – normal distributions were chosen for convenience.

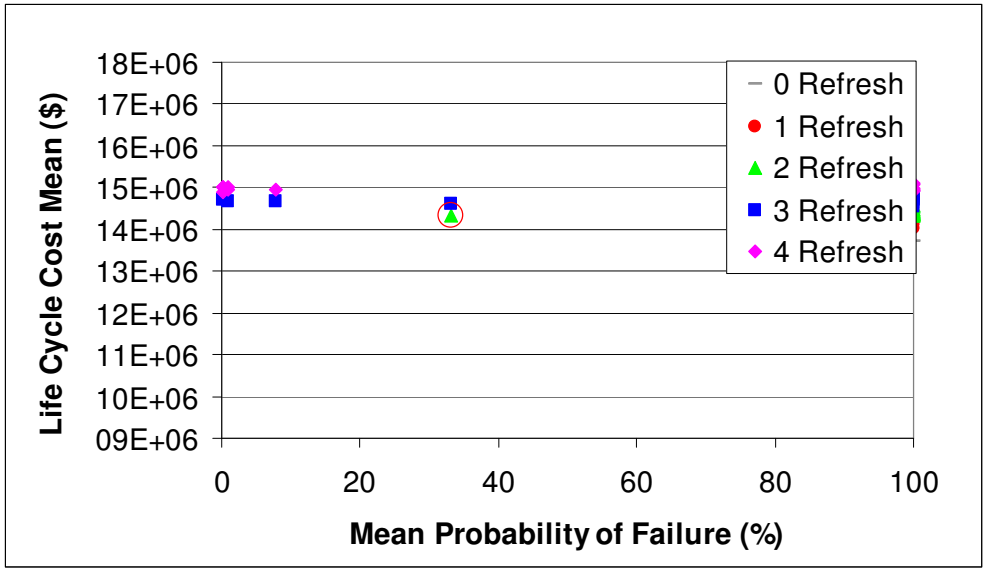


Figure 5-3: MOCA generated refresh plans probability of failure versus life-cycle cost for the case study system with the application of the generated constraints and statistical parameters accounting for uncertainty. The best refresh plan circled in red (i.e., two refresh plan) in Figure 5-3 is also circled in red in this figure. A detailed plot of the boxed solutions is provided in Figure 5-4.

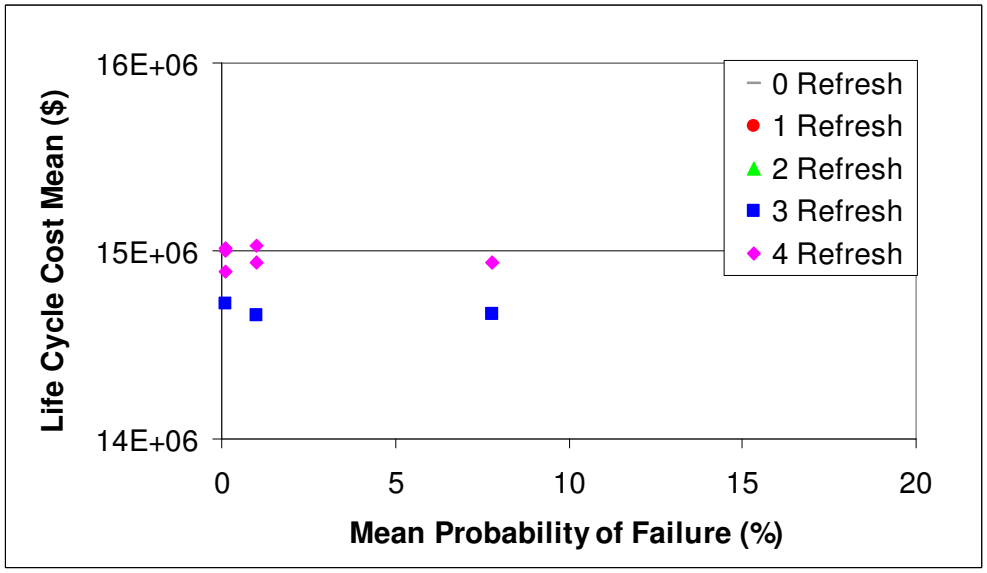


Figure 5-4: MOCA generated refresh plans probability of failure versus life-cycle cost for the case study system with application of the generated constraints and statistical parameters accounting for uncertainty (Close up view of >90% plans).

After running a 1000 samples, the results revealed that the best (i.e., non-violating and least expensive) plan found by deterministic methods is not the best plan (i.e., minimized probability of failure and life cycle cost) when input uncertainty is present. Looking at Figure 5-3, the horizontal-axis is the mean probability of failure, which stays constant throughout time assuming the date distribution parameters (e.g., μ^*, σ^*) do not change. The vertical-axis is the life cycle cost of the system for each refresh plan; however, unlike the previous figures, in Figures Figure 5-3 and Figure 5-4 the vertical-axis is the mean life cycle cost. Figure 5-4 shows the nine refresh plans with a 10% and lower probability of failure. Note, no plans have 0% probability of failure (i.e., 100% probability of satisfying all the constraints when uncertainties are considered).

Uncertainty allows for better model approximation to "real-world" conditions since nothing measured is without uncertainty, and it allows us to be more risk seeking rather than adverse so as to consider refresh plans that have less than 100% probability of satisfying all constraints, which we would otherwise dismiss.

5.2 Medical Laboratory Equipment ("Strong A" Events)

This case study is on automated chemical analyzer system by an In Vitro Diagnostic (IVD) company whose identity will be kept confidential. The chemical analyzer system is classified as an FDA Class I medical device (i.e., non-invasive) and is used in hospitals and laboratories to test several biological specimens in batches for various toxins, bacteria, and other chemicals. The objective of this study was to determine the best design refresh plan.

Because the chemical analyzer system is a medical device, it is currently exempt from the RoHS directive (see Section 5.1), which is a form of environmental legislation. Medical systems like the chemical analyzer system are not expected to remain exempt from RoHS

indefinitely, so new systems sold into the EU may have to conform to RoHS requirements in the near future. A significant population of the chemical analyzer system has already been delivered to the field that are not RoHS compliant and manufacturing of the system will continue through 2016 or longer. The IVD company, would like to impose an internal target date to make the product RoHS compliant in order to prepare for the anticipated end of the applicable exemption.

Since the RoHS directive is a form of legislation then according to the constraint taxonomy in Section 3.1, this constraint is identified to be permission based, which means it can be converted into a penalty function.

The effect of imposing this RoHS directive constraint on the chemical analyzer system is that any non-RoHS compliant components (tin-lead solder finish parts) will effectively be obsolete on the internal RoHS compliance date. Fielded instances of the chemical analyzer system that have non-RoHS compliant components can continue to operate (and be maintained) indefinitely without having to be made RoHS compliant – even after the exemption ends, but new systems cannot be sold in the EU that are not RoHS compliant. The obsolescence events driven by an internal date to become RoHS compliant are "Strong A" because new systems cannot be manufactured with non-RoHS compliant components, and existing systems do not have to be updated (backfitted).

The idea of "effective obsolescence" is important since it calls into question the definition of obsolescence and how it translates into this particular scenario. Obsolescence is defined as the loss or impending loss of original manufacturers of items or suppliers of items or raw materials; but in a more general sense it can mean that access to and subsequent use of an item is no longer possible/permissible beyond a certain point in time. Extending the idea

of obsolescence to mean whether an item is permitted for use requires more information such as who are authority and supplicant actors? In this case, the IVD company is the supplicant and the entire European Union along with anyone who conducts business and enforces the RoHS directive with them (which practically speaking includes every country in the world) are the authority figures. It can be argued that the permission to use (which in the context of this case study means to incorporate in manufactured products and sell) an item can effectively obsolete the item from the IVD company's point of view. Incorporating this idea into the constraint building algorithm means that all non-RoHS compliant components will have their obsolescence dates shifted back to the "effective obsolescence" date, which is the IVD company's internal RoHS compliance date.

The chemical analyzer system consists of 21 subassemblies (also referred to as "boards") within which there are 1,224 components, 794 of which are unique. A total of 204 components are known to be RoHS non-compliant, which means all of those components experience a "Strong A" obsolescence event when the internal RoHS compliance date arrives. Figure 5-5 plots the RoHS status of the components in each subassembly as a percentage of the total number of component instances in each subassembly. Note, for a significant number of components, the RoHS status was not available. It was assumed that if no RoHS status information was found for a component, then the component was RoHS compliant (best case assumption).

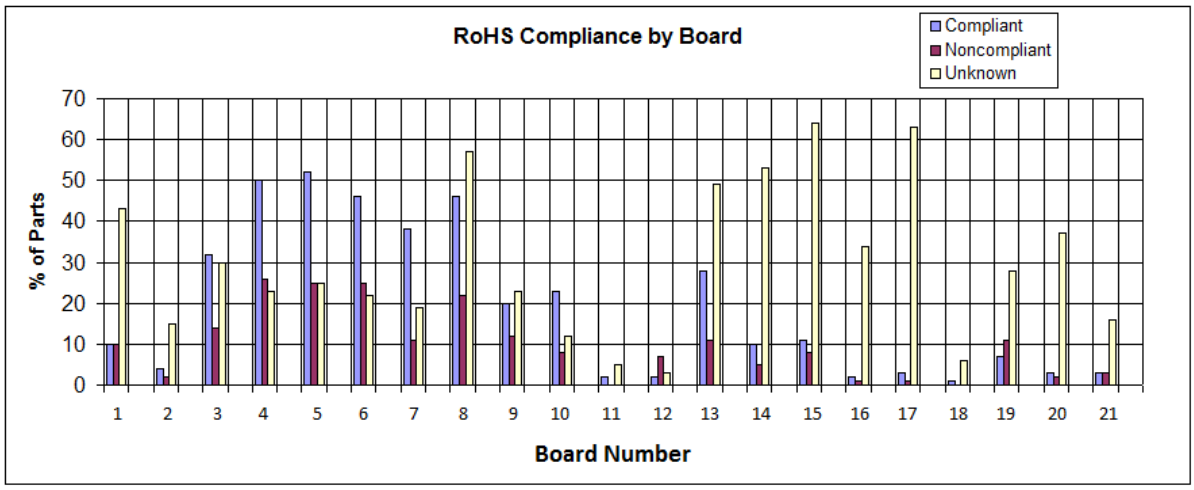


Figure 5-5: RoHS compliance percentage of the chemical analyzer system's 21 subassemblies' total part counts.

The chemical analyzer system's bill of materials, system production schedule, and constraints were input into the DRP process (implemented in the MOCA refresh planning tool). The following parameters were set:

Internal RoHS Compliance Date = January 1, 2014

"Look ahead" time (*LAT*) = 2 years

Valuation Year of Money = 2009

Start of Analysis = January 1, 2009

End of Support = January 1, 2016

Cost of Money = 9.5%

Design Refresh NRE = $50K fixed + $40K/board touched + $1K/part touched

Figure 5-6 shows the results of the chemical analyzer system MOCA model with all generated constraints applied. A good check is to see if the feasible single design refresh plans satisfy the generated constraints by hand. While there are many non-RoHS compliant components that create a "Strong A" obsolescence event and thus require a constraint to be generated, since we are using the internal RoHS compliance date as our "effective

obsolescence" date for all affected components, then ultimately all those generated constraints will be exactly the same, meaning effectively there is just one "Strong A" based temporal constraint for this case study. To generate the constraint by hand, the production schedule is needed since the date of the next production event after the internal RoHS compliant date (i.e., "effective obsolescence date") is the end of the constraint period for this "Strong A" obsolescence event type based constraint.

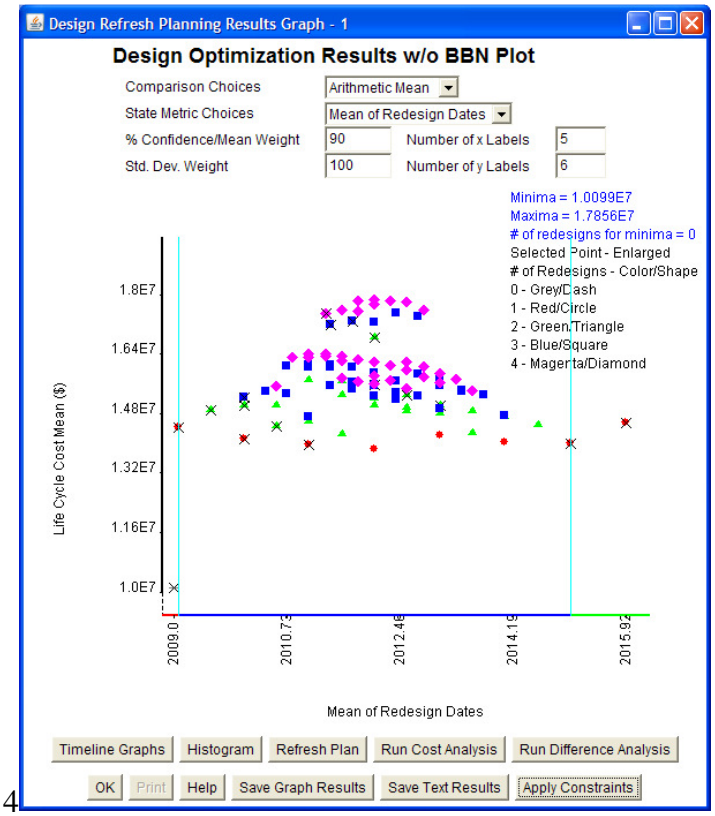


Figure 5-6: MOCA results plot with temporal constraints (based on "Strong A" obsolescence events) applied.

The chemical analyzer systems are manufactured annually and for the case study it was assumed that production for each year began on January 1, which means that the next production event after the "effective obsolescence date" of January 1, 2014 is January 1, 2015. The constraint period start date is January 1, 2012 because the "look-ahead" time is 2 years from the "effective obsolescence event" on January 1, 2014. The constraint period end

date is then January 1, 2015. Referring back to Figure 5-6 and looking at just the single design refresh plans, it can be seen that any plan outside the constraint period is marked with an "X" indicating they have violated the constraint set, which is a good qualitative indicator that the constraint has been properly applied. Note, penalty costs were not yet included in this case study.

The important observation for this case study is that while the "No Refresh" plan has the lowest associated cost and has a cost difference of about 4 million dollars from the lowest single design refresh plan, it does not satisfy the constraint, making it an infeasible design refresh plan.

Figure 5-7 shows solution results for a range of different possible internal RoHS compliance dates. Also included on Figure 5-7 are different possible discount rates on money and different end of support (*EOS*) dates for the system. For the earliest *EOS*

dates, 2013 and 2014, the best solution is to never become RoHS compliant (RoHS compliance date after the *EOS*); for later *EOS* dates the best solution is to delay the internal RoHS compliance date at long as possible if the discount rate is high (9%). However, when the discount rate is low (2%), the best solution is to become RoHS compliant earlier as opposed to later (in 2011).

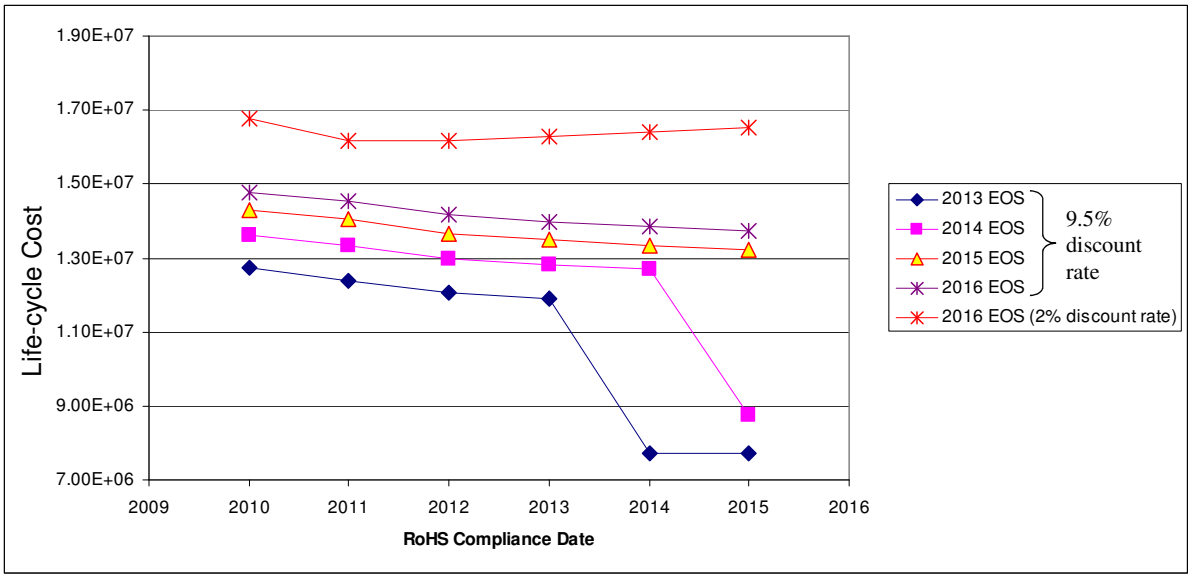


Figure 5-7: Life-cycle cost related to the RoHS internal compliance date.

5.3 Ship-Board Communications System ("Strong B" Event)[20]

This case study is for the Integrated Shipboard Network System (ISNS), specifically the variant AN/USQ-153(V)8, which is a tactical local area network infrastructure to be placed on Mine Countermeasure (MCM) class ships. The ISNS is managed by the Program Management, Warfare (PMW) 160 office, which is part of the Space and Naval Warfare Systems Command (SPAWAR). The objective of this study was to find the best design refresh plan using the design refresh planning process.

What makes this case study so interesting is the system's bill of materials (BOM) consists of many pieces of consumer software, which is unlike the case study described in Section 5.2 whose BOM was primarily made up of hardware. Software obsolescence is similar to hardware obsolescence in terms of purchasing it from the original equipment manufacturer (OEM) known as "end of sale." When the OEM stops selling the item it is obsolete; however, software is fundamentally different from hardware in that software

[20] All information included in this section was taken from the following publically available reference [63]. Reference [63] was approved for public release by the US Navy, PMW 160 in 2008.

requires technical support throughout its life-cycle where as hardware for the most part does not. This is due to the fact that during the development phase, the software developers can only create solutions for only known potential problems or threats against the operation and security of the software. Any future problems or threats encountered in the software's operating environment would need to be resolved in the future using software updates, patches, etc. More to the point, in the case of the ISNS, software to software and software to hardware connectivity can be significantly impacted if technical support from the software developers ends, which for this mission critical system is not acceptable. It is for this reason that many software components in the ISNS BOM are identified as causing "Strong B" type obsolescence event.

The AN/USQ-153(V)8 variant of the ISNS has 52 software components (10 of which are identified as causing a "Strong" type obsolescence event) and 27 hardware components. There were 4 "Strong A" and 5 "Strong B" constraints. The AN/USQ-153(V)8 bill of materials, system production schedule (Table 5-5), and constraints were input into the DRP process (implemented in a Visual Basic Applications (VBA) program external of MOCA). The following parameters were set:

"Look ahead" time (*LAT*) = 2 years

Valuation Year of Money = 2000

Start of Analysis = January 1, 2000

End of Support = January 1, 2015

Cost of Money = 4%

Table 5-5: Production schedule for the AN/USQ-153(V)8.

//year	month	quantity	kind	item	name	spares	end date
2007	0	4	Reorder	board	V8-1-full		2015
2008	0	4	Reorder	board	V8-1-full		2015
2009	0	4	Reorder	board	V8-1-full		2015
2010	0	2	Reorder	board	V8-1-full		2015

Figure 5-8 is a MOCA plot of the AN/USQ-153(V)8 results without generating constraints. The key observations for this result is that the zero design refresh plan is the best plan out of all candidate design refresh plans and there are 16 candidate design refresh plans not including the zero design refresh plan.

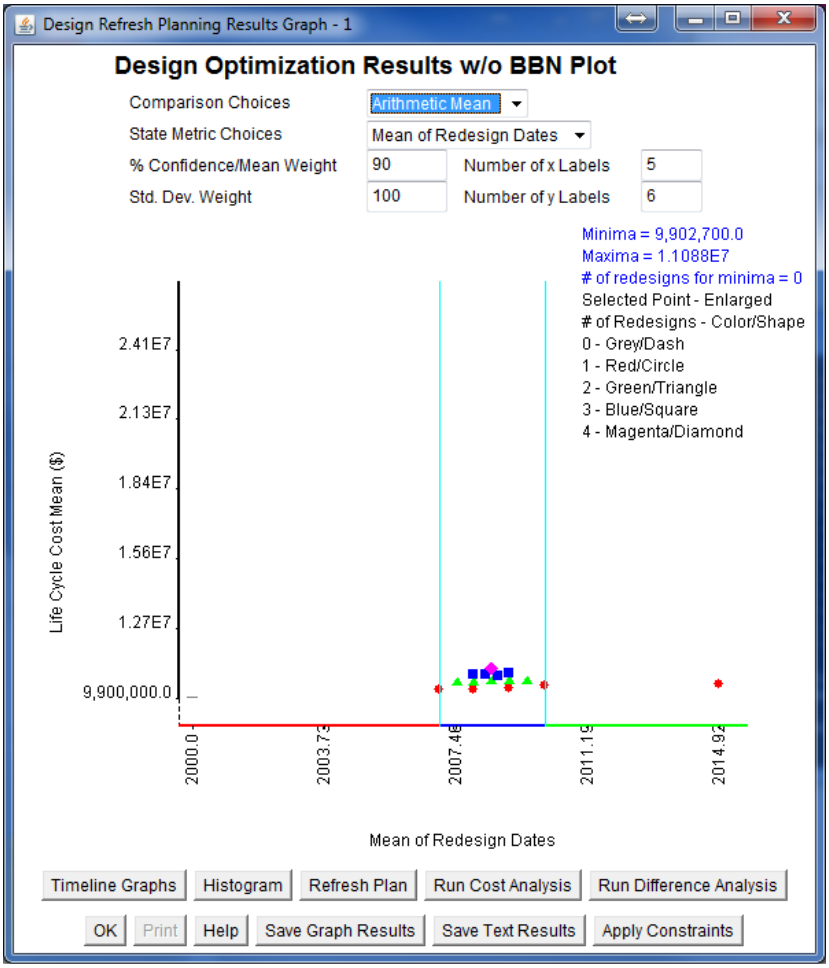


Figure 5-8: MOCA results for ISNS with "Strong B" constraints not generated.

Next an external VBA program was used to generate the temporal constraints for the AN/USQ-153(V)8 system and the backfit production schedule for the "Strong B" constraints

(Table 5-6). A backfit is an implementation of a design refresh to fielded systems. For example the design refresh that completes before the January 1, 2008 system production event shown in Table 5-5 is implemented to the fielded systems built in 2007 and it is also implemented for the about to be built systems in 2008, which can be seen in the quantity of backfits produced shown in the first row of Table 5-6.

Table 5-6: "Strong B" constraint backfit schedule.

//year	month	quantity	kind	item	name	spares	end date
2008	6	8	Reorder	board	V8-retrofit		2015
2009	6	12	Reorder	board	V8-retrofit		2015
2009	6	12	Reorder	board	V8-retrofit		2015
2014	3	14	Reorder	board	V8-retrofit		2015
2010	6	14	Reorder	board	V8-retrofit		2015
2010	6	14	Reorder	board	V8-retrofit		2015

Figure 5-9 shows the effect of generating and applying the “Strong” obsolescence type event constraints. Notice the increase in the number of candidate design refresh plans from 16 to 163. The dramatic increase of candidate design refresh plans is due to the inserted system production events used to implement the backfits that came from the “Strong B” constraints. The main takeaway from this case study is that after generating and applying the constraints the best design refresh plan is a three refreshes design refresh plan, which is a 14 million dollar difference from the best plan without generating and applying the constraints.

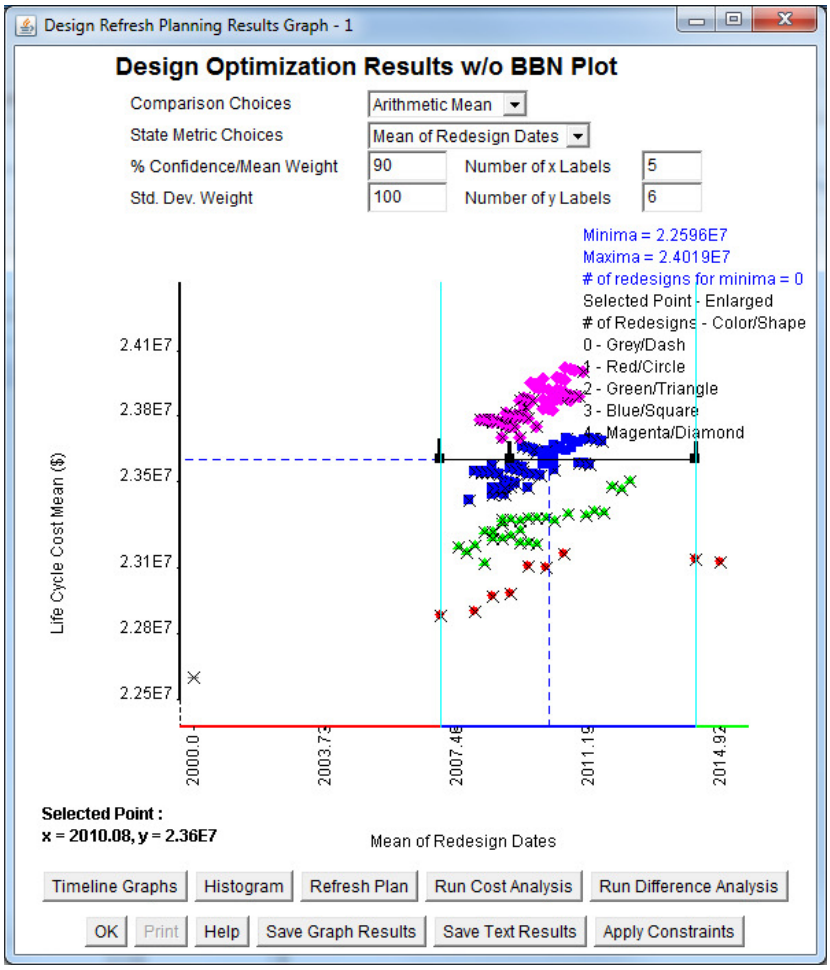


Figure 5-9: MOCA results for ISNS with "Strong B" constraints generated and applied.

Chapter: 6 Contributions, Conclusions and Future Work[21]

This dissertation developed a new refresh planning model that satisfies refresh structure requirements (i.e., requirements that constrain the form of the refresh plan to be periodic) and developed the definition, generalization, synthesis and application of part-unique temporal constraints in the design refresh planning (DRP) process for systems impacted by DMSMS-type obsolescence.

Periodic refresh plans are required by applications that are refresh deployment constrained such as ships and submarines. The new refresh planning model developed in this dissertation has been shown to require less data and execute faster than the existing state-of-the-art discrete event simulation solutions for problems where a periodic refresh solution is required.

Temporal constraints on all types of DRP problems have also been developed. The basis for each temporal constraint was assumed to be in an implicit form, meaning taken by itself the basis of the temporal constraint could not be directly applied to the DRP process and required a method of translation into an explicit form that could be directly applied to the design refresh plans. The methodology created uses a constraint taxonomy and obsolescence event type definitions as a means of translating the basis of a temporal constraint from an implicit form into an explicit form. The treatment and application of these temporal constraints is complementary to the work presented in [64] that incorporates technology roadmapping information into the DRP process.

The constraint generating methodology was implemented in the MOCA DRP tool [65] and has been applied to several real systems including a chemical analyzer system by an

[21] For a list of publications associated with the work presented in this dissertation see Appendix D.

In Vitro Diagnostic (IVD) company and an Integrated Shipboard Network System (ISNS) for the Navy's Mine Countermeasure (MCM) class ships.

6.1 Contributions

Prior to the work performed in this dissertation, design refresh planning (DRP) methodologies did not have a means of formulating, generating, or handling either structural (solution form) or temporal constraints. The specific contributions to the design refresh planning field include:

- Created the first design refresh planning solution that finds the optimum periodic refresh solution – previous solutions find either single refresh optima or global non-periodic optima. For the special case of periodic refreshes the new solution is 50% faster and requires 50% less data input than state-of-the-art global refresh solutions.
- Creation of fundamental obsolescence event type definitions used to translate imposed temporal constraints in an implicit form (i.e., a constraint that is not in terms of the design refresh variable) into an explicit form (i.e., a constraint that can be directly applied to the design refresh variable).
- Creation of an algorithm for synthesizing temporal constraints from obsolescence event type definitions that enables automated temporal constraint generation and application within design and system management planning tools. Creation of constraint taxonomy for the obsolescence management of systems which can be used to assess the convertibility of a constraint into a penalty function.

This dissertation postulates that by incorporating temporal constraints into the DRP process, the resulting best solution will result in minimization of the product's life-cycle cost

by minimizing constraint violation and thus minimizing penalties incurred as a result of violating a constraint. The taxonomy, definitions, algorithms, and methods developed in this dissertation can be used in any design refresh planning methodology and is not limited to any particular implementation. The temporal constraint building methodology developed is a means of modeling a variety of constraints that restrict the DRP process temporally but are not in a form readily translatable into the design variables and parameters used in the DRP process.

6.2 The Application Scope of this Work

The application of this work is aimed at improving the ability to manage mission, infrastructure and safety critical systems that must be supported for long periods of time with constituent components whose supply chain is not controlled by the system manufacturer/sustainer. The demand for managing these types of systems is increasing and likely to become a more central concern as systems age – if the money is not available to buy new systems, you will have to learn to take better care of the old systems [66].

As an example of the application scope, consider an organization manufacturing and managing a fleet of legacy systems.[22] The budget for sustaining these systems is limited; however, system availability and security must be maximized, which means while reactive approaches to mitigating obsolescence problems may ensure system availability and be less expensive, some design refreshes are required to ensure system security. These legacy systems are sustainment dominated meaning the cost of maintaining them and guaranteeing a

[22] A legacy system is a system that continues to be used, typically because it still functions for the users' needs, even though newer technology or more efficient methods of performing the task that the system performs are available [67] [68]. Legacy systems may not be replaced by newer systems for long periods of time because they are generally very expensive to replace and qualify. Common examples of legacy systems include: FAA radar systems, military systems, 911 systems for cities, traffic light control systems, industrial controls, power plant control system, and other safety, infrastructure and mission critical systems.

specific level of system availability is more expensive than the system's initial procurement costs. Constraints that affect the sustainment of these systems can come in different forms and from different sources, and usually change over time. Legislations by various governments that were enacted in just the past 10 years have resulted in a dramatic change in how systems can be maintained and with what components as well as the cost to maintain them. While many legacy systems may be excluded from legislations that regulate content and waste in manufactured products, they do not control the supply chain of the constituent components used in their systems and are therefore indirectly impacted as the products/systems that do control the supply chains are subject to the legislation. If constraints are not applied properly or if they are not applied at all, there is a risk that the system will be vulnerable, e.g., a tactical communications network being hacked due to obsolete programs or operating systems. Another associated risk is that the system will be burdened with unexpected costs, e.g., added costs incurred to have newer or compliant components reverted to match older requirements. For design refresh planning to be advantageous, constraints need to be incorporated into the process.

6.3 Future Work

The objective of this dissertation is to provide a means for incorporating temporal constraints into the DRP process; however, while the methodology developed in this dissertation is a guideline for achieving this objective, it is recognized that not all constraints imposed on the DRP process can be represented as a temporal constraint. It is also not this dissertation's goal to improve or change the DRP process. The generation and application of temporal constraints is intended for the DRP process but the ideas and methods used to create them are not limited to it.

6.3.1 Incorporating Holding Cost into the MpMe Model

The derivation of the MpMe model assumed the holding costs were sunk (i.e., costs have already been paid); however, in actuality this is not always the case. Holding cost for this dissertation is derived with the following assumptions. The holding cost is:

1) Paid at the end of one year intervals starting from year zero.
2) Dependent on the component price.
3) Proportional to the inventory (i.e., there are no fixed costs).

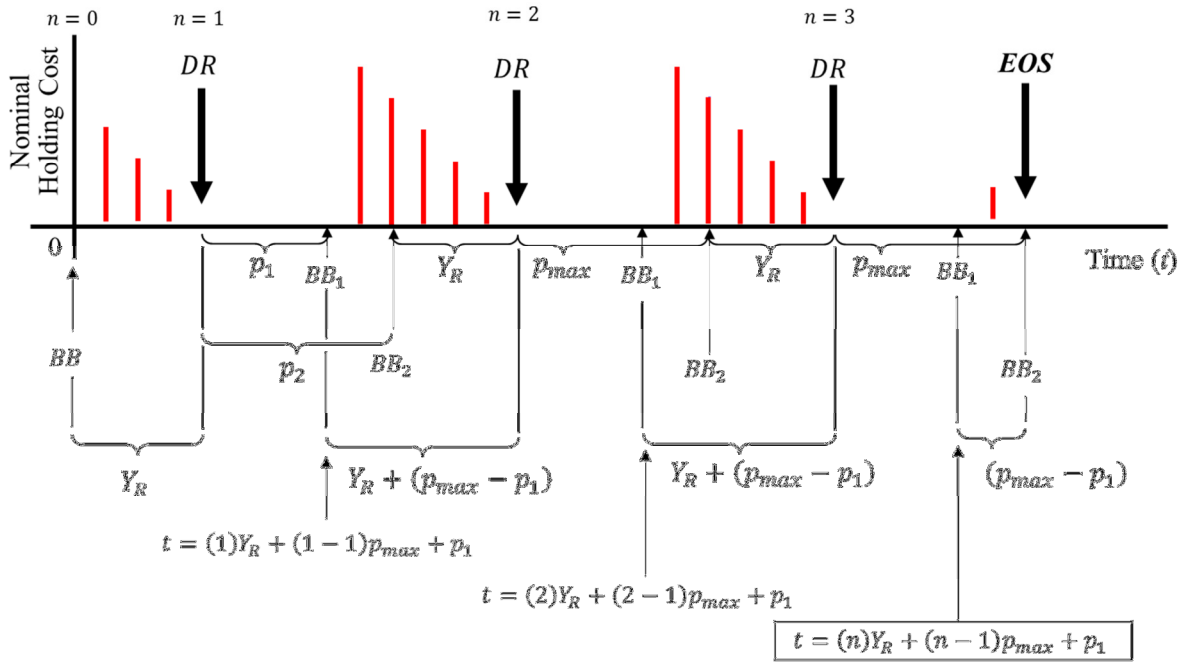


Figure 6-1: Diagram of the nominal holding costs for bridge buys within the MpMe model

Figure 6-1 illustrations how holding cost is modeled in this dissertation. The horizontal axis is time and the vertical axis is the nominal holding cost (i.e., the effects of inflation and interest are not applied). Figure 6-1 is a bar graph, where each red bar represents the nominal holding cost paid at time t. Notice that at each bridge buy (BB) there is no holding cost. This is because for this dissertation holding cost is paid at the end of each year. The time at which a bridge buy is made can be related to the waiting time (Y_R), the

design refresh period (n), the maximum component procurement life (p_{max}) and the component procurement life (p_1).

$$t = (n)Y_R + (n-1)p_{max} + p_1 \qquad \textbf{Equation 6-1}$$

Similar to the derivation of the bridge buy cost terms done in Section 2.2.4, there are three terms that make up the holding cost. This is because looking at Figure 1-1, there are three distinct time periods that will yield different demand quantities. The first, second and third time periods have durations of time Y_R, $Y_R + (p_{max} - p_1)$ and $(p_{max} - p_1)$, respectively. The only remaining piece of information needed to derive the holding cost terms is the cost paid to store a component for one year. This relationship will be referred to in this dissertation as the storage fraction and has the notation M_{sf}, and it is the fraction of the component's price paid to store one instance of the component for one year. Using the ideas previously discussed, the following three terms that make up the holding cost have been created:

$$TC_{HC}$$

$$= \sum_{n=0}^{0}\left(\sum_{t=1}^{Y_R}\left(\frac{P_{0_i}M_{sf}Q_i(Y_R - t + 1)}{(1+r)^t}\right)\right)$$

$$+ \sum_{n=1}^{N-1}\left(\sum_{t=1}^{Y_R+(p_{max}-p_i)}\left(\frac{P_{0_i}M_{sf}Q_i(Y_R + (p_{max}-p_i) - t + 1)}{(1+r)^{(n)Y_R+(n-1)p_{max}+p_i+t}}\right)\right) \qquad \textbf{Equation 6-2}$$

$$+ \sum_{n=N}^{N}\left(\sum_{t=1}^{p_{max}-p_i}\frac{P_{0_i}M_{sf}Q_i\big((p_{max}-p_i) - t + 1\big)}{(1+r)^{(n)Y_R+(n-1)p_{max}+p_i+t}}\right)$$

An important caveat that needs to be mentioned is that summations are done using natural (integer) numbers as the incremental value; however, Y_R, $Y_R + (p_{max} - p_1)$ and $(p_{max} - p_1)$ may result in any real (i.e., decimal) number. Because of this, the following manipulation needs to be made. Any real number can be broken down into two parts: an integer (*int*) and a fraction (*fr*). Let Y_R be a real number such that:

$$Y_R = int(Y_R) + fr(Y_R) \qquad \textbf{Equation 6-3}$$

If we are to assume that all real numbers have this property then any summation with the incremental value that is not guaranteed to be a natural (i.e., integer) number then we can make the following equivalency:

$$\sum_{t=1}^{Y_R} a^t = \sum_{t=1}^{int(Y_R)} a^t + \sum_{t=Y_R}^{Y_R} a^t fr(Y_R) \qquad \textbf{Equation 6-4}$$

Using ideas and conventions shown in Equation 6-3 and Equation 6-4 we can now re-write Equation 6-2.

$$TC_{HC} = \sum_{n=0}^{0}\left(\sum_{t=1}^{int(Y_R)}\left(\frac{P_{0_i}M_{sf}Q_i(Y_R - t + 1)}{(1+r)^t}\right) + \sum_{t=Y_R}^{Y_R}\left(\frac{P_{0_i}M_{sf}Q_i(Y_R - t + 1)fr(Y_R)}{(1+r)^t}\right)\right)$$

$$+\sum_{n=1}^{N-1}\left(\sum_{t=1}^{int(Y_R+(p_{max}-p_i))}\left(\frac{P_{0_i}M_{sf}Q_i(Y_R + (p_{max} - p_i) - t + 1)}{(1+r)^{(n)Y_R+(n-1)p_{max}+p_i+t}}\right)\right.$$

$$\left.+\sum_{t=Y_R+(p_{max}-p_i)}^{Y_R+(p_{max}-p_i)}\left(\frac{P_{0_i}M_{sf}Q_i(Y_R + (p_{max} - p_i) - t + 1)fr(Y_R + (p_{max} - p_i))}{(1+r)^{(n)Y_R+(n-1)p_{max}+p_i+t}}\right)\right) \quad \textbf{Equation 6-5}$$

$$+\sum_{n=N}^{N}\left(\sum_{t=1}^{int(p_{max}-p_i)}\left(\frac{P_{0_i}M_{sf}Q_i\big((p_{max} - p_i) - t + 1\big)}{(1+r)^{(n)Y_R+(n-1)p_{max}+p_i+t}}\right)\right.$$

$$\left.+\sum_{t=p_{max}-p_i}^{p_{max}-p_i}\left(\frac{P_{0_i}M_{sf}Q_i\big((p_{max} - p_i) - t + 1\big)fr(p_{max} - p_i)}{(1+r)^{(n)Y_R+(n-1)p_{max}+p_i+t}}\right)\right)$$

Equation 6-5 is in a power sum form of the holding cost term and needs to be put into a form without summations. In order to do this the relationship shown in Equation 6-6 will be used:

$$\sum_{k=0}^{n} kx^k = \frac{x - (n+1)x^{n+1} + nx^{n+2}}{(x-1)^2} \quad \textbf{Equation 6-6}$$

Future work will refine this calculation and incorporate it with the MpMe model. Additional model verification will be done using the MOCA model with a non-zero holding cost.

6.3.2 Periodicity with respect to Optimality

This dissertation is based on the idea of a solution requirement that forces the design refresh planning model to produce only periodic design refresh plans; however, the question of where periodic design refresh plans are optimal regardless of a solution requirement is

something worth investigating. What conditions are necessary for a periodic refresh plan to be optimal over a non-periodic refresh plan? The first attempt to explore this concept is shown on Table 6-1 which is a table of *PSTD* values (i.e., a measure of a design refresh plan's periodicity where *PSTD*=0.0 means the design refresh plan is periodic and *PSTD*>0 means the plan is non-periodic) for various design refresh cost values. The optimum number of refreshes per refresh plan is indicated by the color green so for example in the case where the design refresh cost is $1, the optimal number of refreshes per refresh plan is 6 and that optimal plan with 6 refreshes had a *PSTD* value of 4.0 indicating it was a non-periodic refresh plan. This table shows that as the design refresh cost is increased past $17, the optimum shifts from a non-periodic refresh plan to a periodic one. Note: Case 1 and 2 input data are used for this analysis.

Table 6-1: Table of *PSTD* values with respect to design refresh cost and the number of refreshes per refresh plan

Number of Refreshes in Plan \ Design Refersh Cost [$]	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
6	4.0	4.0	5.7	5.4	5.4	5.4	4.5	2.7	2.3	2.3	0.9	0.9	0.9	0.9	2.5	2.5	2.5	4.7	4.7	4.7
5	6.5	6.5	6.5	6.5	6.5	4.8	4.8	4.8	3.3	0.0	1.0	1.0	1.0	0.0	1.5	1.5	1.5	2.5	2.5	2.5
4	7.0	7.0	7.0	5.6	5.6	5.6	5.6	5.6	2.9	2.9	3.6	3.6	1.2	1.2	2.5	2.5	2.5	0.0	0.0	0.0
3	7.1	7.1	4.2	4.2	4.2	4.2	4.2	4.2	4.2	2.8	2.8	2.8	2.8	2.8	0.0	0.0	0.0	0.0	0.0	0.0

Number of Refreshes in Plan \ Design Refersh Cost [$]	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
6	4.7	4.7	4.7	4.7	5.7	5.7	5.7	5.7	5.7	5.7	5.7	5.7	5.7	5.7	5.7	5.7	5.7	5.7	5.7	5.7	5.7
5	2.5	2.5	2.5	4.8	5.9	5.9	5.9	5.9	5.9	5.9	5.9	5.9	5.9	5.9	5.9	5.9	5.9	5.9	5.9	5.9	5.9
4	0.0	0.0	0.0	3.1	3.1	3.1	3.1	3.1	3.1	3.1	3.1	3.1	3.1	3.1	3.1	3.1	3.1	3.1	3.1	3.1	3.1
3	0.0	0.0	2.1	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0

What should be noted from Figure 6-2 is that best design refresh plan for a given number of refreshes per refresh plan tend to go from non-periodic to periodic an then back to being non-periodic as the design refresh cost is increased from $1 to $40.

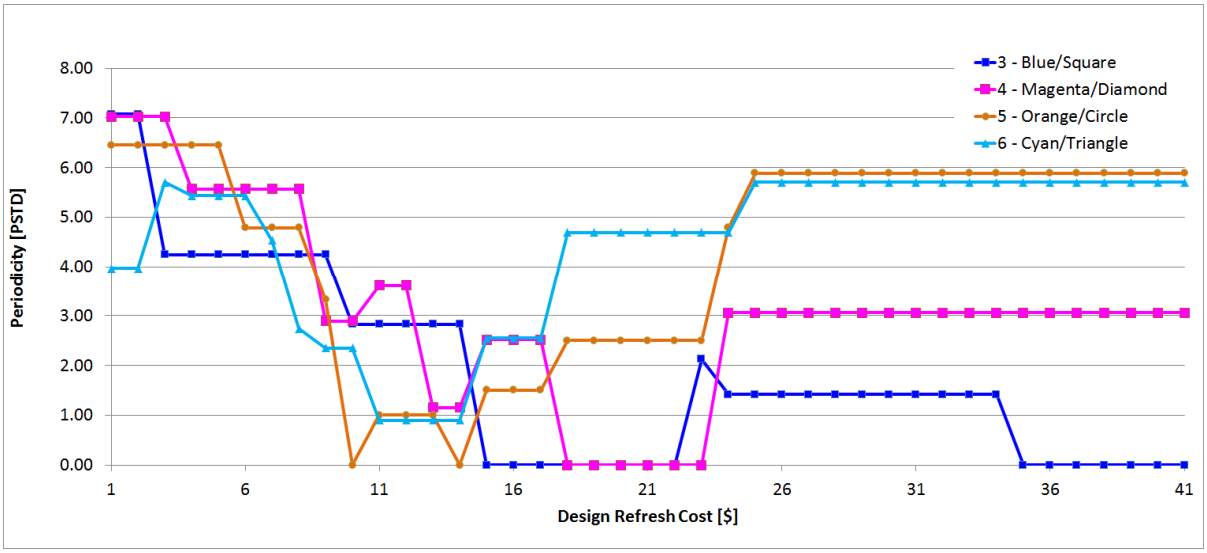


Figure 6-2: This is a plot of *PSTD* values for different numbers of refreshes per plan with respect to the design refresh cost

Future research would define the key parameters that drive the optimal solution to be a periodic and offer further insight as to why this occurs.

6.3.3 Reward Functions

This dissertation created a method of generating temporal constraints and in some cases converted constraints into conditional penalty step functions that added a fixed penalty cost to the life-cycle cost associated with a design refresh plan that violated the constraint. Future work could investigate the potential for conditional incentive step functions rather than conditional penalty step functions. For example, some governments provide tax incentives for inserting qualified environmentally friendly technology within a specified period of time [69] [70] [71]. If conditional incentive step functions are created then additional future work could combine the probabilities for satisfying a constraint for a given design refresh plan, the conditional penalty and incentive step functions and construct a decision-tree based model used to calculate an expected life-cycle value for each design refresh plan.

6.3.4 Non-hardware/software obsolescence

Other key elements for supporting systems can become obsolete in addition to hardware and software. These elements include people (skills) and intellectual property. Skill obsolescence, in the context of this dissertation, means the inability of an organization that supports a system to obtain people with the necessary skill set to execute the required support activities, i.e., the skill to run a particular piece of test equipment, perform failure analysis on relevant hardware, troubleshoot a system, or fix software bugs in software developed in a particular language [72] [73]. In addition, for technology items, the intellectual property associated with making, fielding, and/or supporting systems may be a relevant concern. For example, intellectual property used to design application specific integrated circuits (ASICs) may be licensed for only a finite period of time and when the licenses expire, the ability of an organization to have replacement ASICs made may terminate.

6.3.5 Constraint Violation Costing Problem Definition

More detailed models for constraint violation could be developed. The following more detailed modes were not pursued within this dissertation because the data necessary to populate them was not available. The following is a breakdown of the associated cost of violating a constraint into penalty fees, missed revenue, and production delay expenditures.

$$CVC = PDC + PC + MR$$

Equation 6-7

CVC - Constraint Violation Cost
PDC - Production Delay
PC - Penalty
MR - Missed Revenue

$$PDC = \left(LR + EOR + FOR\right)dt$$

Equation 6-8

where,

PDC - Production Delay Cost
LR - Labor Rate [$/hour]
EOR - Equipment Overhead Rate [$/hour]
FOR - Facility Overhead Rate [$/hour]
dt - Production Delay Time [hour]

$$PC = NRP + RP$$

Equation 6-9

where,

PC - Penalty Cost
NRP - Non-Reoccurring Penalty
RP - Reoccurring Penalty

$$MR = (PRR)dt$$

Equation 6-10

where,

MR - Missed Revenue
PRR - Projected Revenue Rate [$/hour]
dt - Production Delay Time [hour]

Appendix A: Geometric Series Reduction

This appendix shows the derivations used to create Equation 2-11 through Equation 2-14.

Bridge Buy Term:

$$C_{BB} = \sum_{n=1}^{N} \frac{P_0 Q Y_R}{(1+r)^{(n-1)(Y_R+p)}}$$

Bring all constants to the front of the summation.

$$C_{BB} = P_0 Q Y_R \sum_{n=1}^{N} \frac{1}{(1+r)^{nY_R+np-Y_R-p}} = P_0 Q Y_R \sum_{n=1}^{N} \frac{(1+r)^{Y_R}(1+r)^{p}}{(1+r)^{nY_R}(1+r)^{np}}$$

Put the summation in the geometric series form.

$$C_{BB} = P_0 Q Y_R (1+r)^{(Y_R+p)} \sum_{n=1}^{N} \frac{1}{(1+r)^{n(Y_R+p)}} = P_0 Q Y_R (1+r)^{(Y_R+p)} \sum_{n=1}^{N} (1+r)^{-n(Y_R+p)}$$

Create a geometric series analogous equation to simplify terms.

$$S_N = C_1 \sum_{n=1}^{N} {C_2}^{nC_3}$$

$$C_1 = P_0 Q Y_R (1+r)^{(Y_R+p)}$$

$$C_2 = 1 + r$$

$$C_3 = -(Y_R + p)$$

Expand summation to reveal summation pattern.

$$S_N = C_1 \sum_{n=1}^{N} {C_2}^{nC_3} = C_1\left({C_2}^{1C_3} + {C_2}^{nC_3} + {C_2}^{nC_3} + \cdots + {C_2}^{NC_3}\right)$$

Bring constants over to left side. Consider this equation A.

$$\left(\frac{S_N}{C_1}\right) = \left({C_2}^{1C_3} + {C_2}^{2C_3} + {C_2}^{3C_3} + \cdots + {C_2}^{NC_3}\right)$$

Multiply both sides by ${C_2}^{1C_3}$. Consider this equation B.

$${C_2}^{1C_3}\left(\frac{S_N}{C_1}\right) = {C_2}^{1C_3}\left({C_2}^{1C_3} + {C_2}^{2C_3} + {C_2}^{3C_3} + \cdots + {C_2}^{NC_3}\right) = \left({C_2}^{2C_3} + {C_2}^{3C_3} + {C_2}^{4C_3} + \cdots + {C_2}^{(N+1)C_3}\right)$$

Subtract equation B from equation A.

$$\left(1 - {C_2}^{1C_3}\right)\left(\frac{S_N}{C_1}\right) = \left({C_2}^{1C_3} + {C_2}^{nC_3} + {C_2}^{nC_3} + \cdots + {C_2}^{NC_3}\right) - \left({C_2}^{2C_3} + {C_2}^{3C_3} + \cdots + {C_2}^{NC_3} + {C_2}^{(N+1)C_3}\right)$$

$$\left(1 - {C_2}^{1C_3}\right)\left(\frac{S_N}{C_1}\right) = {C_2}^{1C_3} - {C_2}^{(N+1)C_3}$$

Solve for S_N and simplify.

$$S_N = \frac{C_1\left({C_2}^{1C_3} - {C_2}^{(N+1)C_3}\right)}{\left(1 - {C_2}^{1C_3}\right)}$$

$$S_N = \frac{C_1\left({C_2}^{1C_3} - {C_2}^{NC_3}{C_2}^{1C_3}\right)}{\left(1 - {C_2}^{1C_3}\right)}$$

$$S_N = \frac{C_1{C_2}^{1C_3}\left(1 - {C_2}^{NC_3}\right)}{\left(1 - {C_2}^{1C_3}\right)}$$

Insert corresponding terms to bring back equation into actual form.

$$C_{BB} = P_0 Q Y_R (1+r)^{(Y_R+p)}(1+r)^{-(Y_R+p)}\frac{\left(1-(1+r)^{-N(Y_R+p)}\right)}{\left(1-(1+r)^{-(Y_R+p)}\right)}$$

$$C_{BB} = P_0 Q Y_R \frac{\left(1-(1+r)^{-N(Y_R+p)}\right)}{\left(1-(1+r)^{-(Y_R+p)}\right)}$$

Design Refresh Term:

The process described in the previous section is the same for this next section.

$$C_{DR} = \sum_{n=1}^{N}\frac{C_{DR_0}}{(1+r)^{(nY_R+(n-1)p)}} = C_{DR_0}\sum_{n=1}^{N}\frac{1}{(1+r)^{(nY_R+np-p)}}$$

$$C_{DR} = C_{DR_0}\sum_{n=1}^{N}\frac{1}{(1+r)^{(nY_R+np-p)}} = C_{DR_0}\sum_{n=1}^{N}\frac{(1+r)^p}{(1+r)^{n(Y_R+p)}}$$

$$C_{DR} = C_{DR_0}(1+r)^p \sum_{n=1}^{N} \frac{1}{(1+r)^{n(Y_R+p)}} = C_{DR_0}(1+r)^p \sum_{n=1}^{N} (1+r)^{-n(Y_R+p)}$$

$$S_N = C_1 \sum_{n=1}^{N} {C_2}^{nC_3}$$

$$C_1 = C_{DR_0}(1+r)^p$$

$$C_2 = 1+r$$

$$C_3 = -(Y_R+p)$$

(*See previous work*)

$$S_N = \frac{C_1 {C_2}^{1C_3}\left(1-{C_2}^{NC_3}\right)}{\left(1-{C_2}^{1C_3}\right)}$$

$$C_{DR} = C_{DR_0}(1+r)^p (1+r)^{-(Y_R+p)} \frac{\left(1-(1+r)^{-N(Y_R+p)}\right)}{\left(1-(1+r)^{-(Y_R+p)}\right)}$$

$$C_{DR} = C_{DR_0}(1+r)^p (1+r)^{-Y_R}(1+r)^{-p} \frac{\left(1-(1+r)^{-N(Y_R+p)}\right)}{\left(1-(1+r)^{-(Y_R+p)}\right)}$$

$$C_{DR} = C_{DR_0}(1+r)^{-Y_R} \frac{\left(1-(1+r)^{-N(Y_R+p)}\right)}{\left(1-(1+r)^{-(Y_R+p)}\right)}$$

$$C_{DR} = C_{DR_0}(1+r)^{-\left(\frac{1}{f}-p\right)} \frac{\left(1-(1+r)^{-EOS}\right)}{\left(1-(1+r)^{-\frac{1}{f}}\right)}$$

Bridge Buy Term:

The three bridge buy terms here come from the fact that in Figure 2-2 there are three distinct time periods with different durations: Y_R, $(Y_R + p_{max} - p_1)$ and $(p_{max} - p_1)$.

$$C_{BB} = \sum_{n=1}^{1} \left(\frac{P_0 Q Y_R}{(1+r)^{(n-1)}}\right) + \sum_{n=2}^{N} \left(\frac{P_0 Q (Y_R + p_{max} - p_1)}{(1+r)^{Y_R(n-1)+p_{max}(n-2)+p_1}}\right) + \frac{P_0 Q (p_{max} - p_1)}{(1+r)^{(Y_R+p_{max})N+p_1}}$$

$$C_{BB} = C_{BB_3^1} + C_{BB_3^2} + C_{BB_3^3}$$

$$C_{BB_2^2} = \sum_{n=2}^{N} \frac{P_0 Q (Y_R + p_{max} - p_1)}{(1+r)^{Y_R(n-1)+p_{max}(n-2)+p_1}}$$

$$C_{BB_2^2} = P_0 Q(Y_R + p_{max} - p_1) \sum_{n=2}^{N} \frac{1}{(1+r)^{Y_R n - Y_R + p_{max} n - 2p_{max} + p_1}}$$

$$C_{BB_2^2} = P_0 Q(Y_R + p_{max} - p_1) \sum_{n=2}^{N} \frac{(1+r)^{Y_R}(1+r)^{2p_{max}}}{(1+r)^{Y_R n}(1+r)^{p_{max} n}(1+r)^{p_1}}$$

$$C_{BB_2^2} = P_0 Q(Y_R + p_{max} - p_1)(1+r)^{Y_R}(1+r)^{2p_{max}}(1+r)^{-p_1} \sum_{n=2}^{N} \frac{1}{(1+r)^{Y_R n}(1+r)^{p_{max} n}}$$

$$C_{BB_2^2} = P_0 Q(Y_R + p_{max} - p_1)(1+r)^{Y_R + 2p_{max} - p_1} \sum_{n=2}^{N} \frac{1}{(1+r)^{(Y_R + p_{max})n}}$$

$$C_{BB_2^2} = P_0 Q(Y_R + p_{max} - p_1)(1+r)^{Y_R + 2p_{max} - p_1} \sum_{n=2}^{N} (1+r)^{-(Y_R + p_{max})n}$$

$$S_N = C_1 \sum_{n=2}^{N} {C_2}^{nC_3}$$

$$C_1 = P_0 Q(Y_R + p_{max} - p_1)(1+r)^{Y_R + 2p_{max} - p_1}$$

$$C_2 = (1+r)$$

$$C_3 = -(Y_R + p_{max})$$

$$S_N = C_1 \sum_{n=2}^{N} {C_2}^{nC_3} = C_1\left({C_2}^{2C_3} + {C_2}^{3C_3} + {C_2}^{4C_3} + \cdots + {C_2}^{NC_3}\right)$$

$$\left(\frac{S_N}{C_1}\right) = \left({C_2}^{2C_3} + {C_2}^{3C_3} + {C_2}^{4C_3} + \cdots + {C_2}^{NC_3}\right)$$

$${C_2}^{1C_3}\left(\frac{S_N}{C_1}\right) = {C_2}^{1C_3}\left({C_2}^{2C_3} + {C_2}^{3C_3} + {C_2}^{4C_3} + \cdots + {C_2}^{NC_3}\right) = \left({C_2}^{3C_3} + {C_2}^{4C_3} + {C_2}^{5C_3} + \cdots + {C_2}^{(N+1)C_3}\right)$$

$$\left(1 - {C_2}^{1C_3}\right)\left(\frac{S_N}{C_1}\right) = \left({C_2}^{2C_3} + {C_2}^{3C_3} + {C_2}^{4C_3} + \cdots + {C_2}^{NC_3}\right) - \left({C_2}^{3C_3} + {C_2}^{4C_3} + \cdots + {C_2}^{NC_3} + {C_2}^{(N+1)C_3}\right)$$

$$\left(1 - {C_2}^{1C_3}\right)\left(\frac{S_N}{C_1}\right) = {C_2}^{2C_3} - {C_2}^{(N+1)C_3}$$

$$S_N = \frac{C_1\left({C_2}^{2C_3} - {C_2}^{1C_3}{C_2}^{NC_3}\right)}{\left(1 - {C_2}^{1C_3}\right)} = \frac{C_1 {C_2}^{1C_3}\left({C_2}^{1C_3} - {C_2}^{NC_3}\right)}{\left(1 - {C_2}^{1C_3}\right)}$$

$$C_{BB_2^2} = P_0 Q(Y_R + p_{max} - p_1)(1+r)^{Y_R + 2p_{max} - p_1}(1+r)^{-(Y_R + p_{max})} \frac{\left((1+r)^{-(Y_R + p_{max})} - (1+r)^{-N(Y_R + p_{max})}\right)}{\left(1 - (1+r)^{-(Y_R + p_{max})}\right)}$$

$$C_{BB} = C_{BB_2^1} + C_{BB_2^2}$$

$$C_{BB_2^1} = \sum_{n=1}^{1} \frac{P_0 Q Y_R}{(1+r)^{(n-1)}} = \frac{P_0 Q Y_R}{(1+r)^{(0)}}$$

$$C_{BB} = \frac{P_0 Q Y_R}{(1+r)^{(0)}} + P_0 Q(Y_R + p_{max} - p_1)(1+r)^{Y_R+2p_{max}-p_1}(1+r)^{-(Y_R+p_{max})} \frac{\left((1+r)^{-(Y_R+p_{max})} - (1+r)^{-N(Y_R+p_{max})}\right)}{(1-(1+r)^{-(Y_R+p_{max})})} + \frac{P_0 Q(p_{max}-p_1)}{(1+r)^{(Y_R+p_{max})N+p_1}}$$

$$C_{BB} = P_0 Q Y_R + P_0 Q(Y_R + p_{max} - p_1)(1+r)^{Y_R+2p_{max}-p_1}(1+r)^{-(Y_R+p_{max})} \frac{\left((1+r)^{-(Y_R+p_{max})} - (1+r)^{-N(Y_R+p_{max})}\right)}{(1-(1+r)^{-(Y_R+p_{max})})} + \frac{P_0 Q(p_{max}-p_1)}{(1+r)^{(Y_R+p_{max})N+p_1}}$$

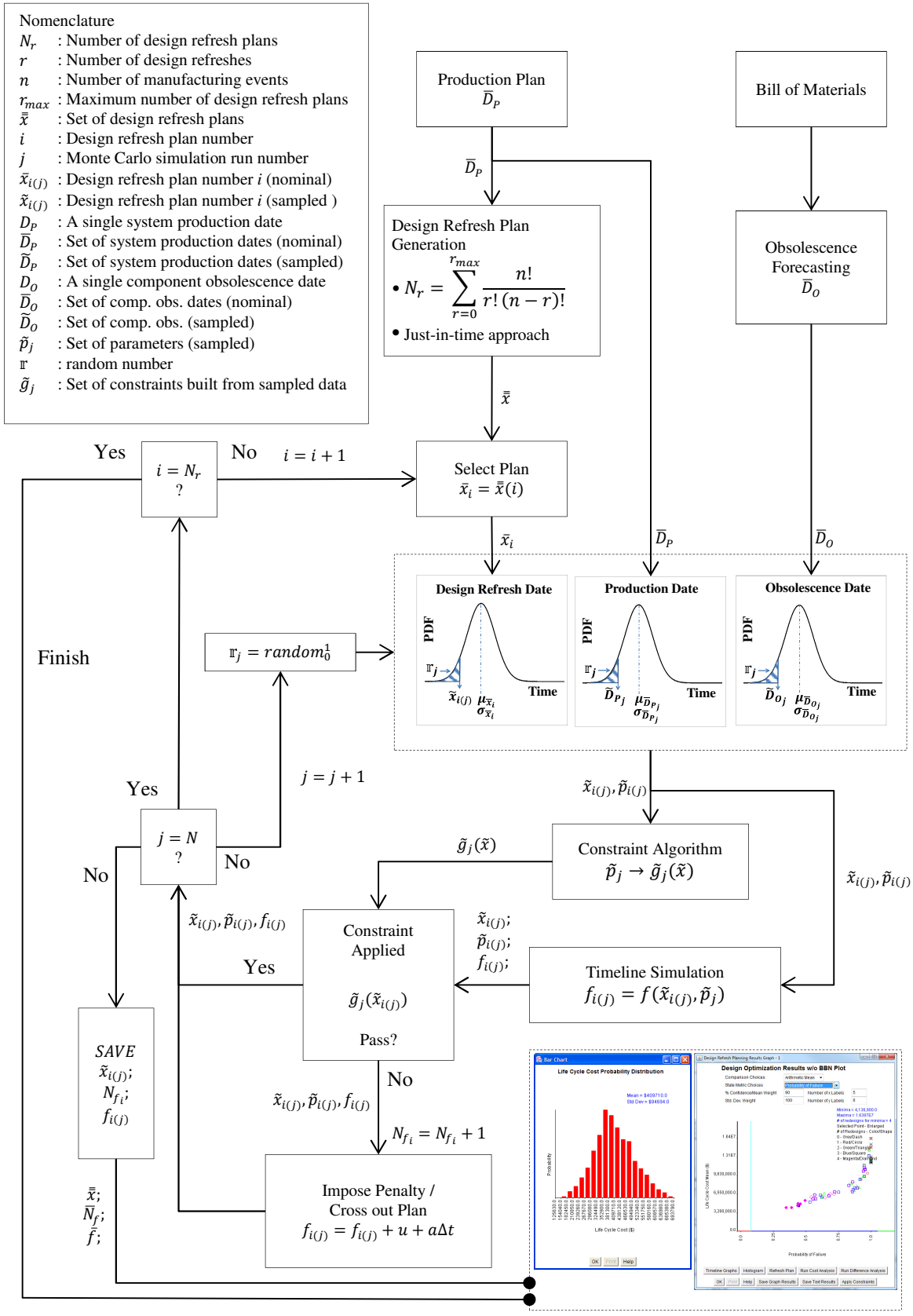
$$C_{BB} = P_0 Q Y_R + P_0 Q(Y_R + p_{max} - p_1)(1+r)^{p_{max}-p_1} \frac{\left((1+r)^{-(Y_R+p_{max})} - (1+r)^{-N(Y_R+p_{max})}\right)}{(1-(1+r)^{-(Y_R+p_{max})})} + \frac{P_0 Q(p_{max}-p_1)}{(1+r)^{(Y_R+p_{max})N+p_1}}$$

$$C_{BB} = P_0 Q Y_R + P_0 Q(Y_R + p_{max} - p_1)(1+r)^{p_{max}-p_1} \frac{\left(1-(1+r)^{-(N-1)(Y_R+p_{max})}\right)}{((1+r)^{(Y_R+p_{max})}-1)} + \frac{P_0 Q(p_{max}-p_1)}{(1+r)^{(Y_R+p_{max})N+p_1}}$$

If a refresh period is defined as $T = Y_R + p_{max}$ and a refresh frequency is defined as $f = \frac{1}{T}$ then the bridge buy term can be re-written in the following way.

$$C_{BB} = P_0 Q\left(\frac{1}{f} - p_{max}\right) + P_0 Q\left(\frac{1}{f} - p_1\right)(1+r)^{p_{max}-p_1} \frac{\left(1-(1+r)^{-\left(EOS-\frac{1}{f}\right)}\right)}{\left((1+r)^{\frac{1}{f}}-1\right)} + \frac{P_0 Q(p_{max}-p_1)}{(1+r)^{\left(\frac{1}{f}\right)N+p_1}}$$

Appendix B: DRP Architecture with Constraint Handling Under Uncertainty



Appendix C: Monte Carlo – Statistical Significance

The Monte Carlo method is a means of modeling uncertainty by sampling values from a distribution or many values from a multitude of distributions each of which could be different in shape, a number of times to get a distribution of results usually in the form of a histogram. While this simple in concept and computationally expensive process is a good solution to analytically infeasible problems, determining the number of trials the Monte Carlo process needs to run in order to get an answer close enough to the actual average is not simple. Two generally acceptable ways to determine the number of Monte Carlo trails required to get an answer distribution close to the actual is empirical and Chebyshev's inequality.

Empirical Method

Based on the Law of Large Numbers [74], performing the same calculation using sampled distributions for inputs a large number of trials will force the average of the results closer to the expected value, and will continue to approach the expected value and become closer as more trials are performed. Plotting the results of the average of a calculation versus the number of times the calculation is repeated (i.e., trials) will illustrate this effect and provide insight as to how many trials are necessary to achieve results that at least appear to be at "equilibrium." Figure A-0-1 demonstrates how the empirical method for finding an acceptable number of Monte Carlo trails to perform.

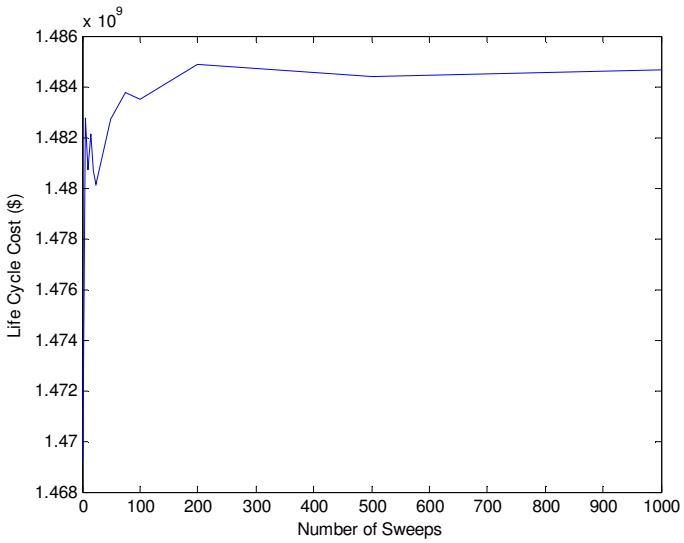


Figure A-0-1: Example of Law of Large Numbers which states that increasing the number of trials or "sweeps" will force the average of the results to approach the expected value.

Chebyshev's Inequality

The idea behind this theory is that for any given sample of a population, for any positive number *k*, the proportion of the data that is contained within *k* standard deviations of the mean is at least:

$$1 - \frac{1}{k^2}$$

Since the Monte Carlo process uses an arithmetic average;

$$A_n = (1/n)(Y_1 + Y_1 \cdots Y_n)$$

then by Chevyshev's Inequality, we have;

$$P(|A_n - \mu|A \geq \epsilon) \leq \frac{\sigma^2}{n\epsilon^2} < \frac{1}{n\epsilon^2}$$

Using this and setting an accuracy (ϵ) and confidence level (p), the number of trials (n) that should be performed is;

$$n \geq \frac{1}{\epsilon^2}(1 - p)$$

Appendix D: Publications Associated with this Work

R. Nelson III and P. Sandborn, "A New Class of Multi-Event Models for Determining the Optimum Refresh Frequency of Systems," accepted for publication *Proceedings of the DMSMS Conference*, New Orleans, LA, August 2012.

L. Zheng, J. Terpenny, P. Sandborn, and R. Nelson III. "Design Refresh Planning Models for Managing Obsolescence," to appear *Proceedings of the ASME International Design Engineering Conferences & Computers and Information in Engineering Conference*, Chicago, Illinois, August 2012.

R. Nelson III and P. Sandborn, "Strategic Management of Component Obsolescence Using Constraint-Driven Design Refresh Planning," to be published *International Journal of Product Life Cycle Management*.

R. Nelson III, P. Sandborn, J. P. Terpenny, and L. Zheng, "Modeling Constraints in Design Refresh Planning," *Proceedings of the ASME International Design Engineering Conferences & Computers and Information in Engineering Conference*, Washington DC, August 2011.

R. Nelson III and P. Sandborn, "Managing Coupled Hardware and Software Obsolescence Using Constraint-Driven Design Refresh Planning," *Proceedings of the DMSMS Conference*, Las Vegas, NV, October 2010.

P. Sandborn and R. Nelson III, "Constraint-Driven Refresh Planning of Systems Subject to Obsolescence," *Proceedings of the DMSMS Conference*, Palm Springs, CA, September 2008.

L. Zheng, R. Nelson, III, J. Terpenny, and P. Sandborn, "Ontology-Based Knowledge Representation for Product Life-Cycle Concepts and Obsolescence Forecasting," *Proceedings of the Industrial Engineering Research Conference*, Reno, NV, May 2011.

L. Zheng, R. Nelson III, J. Terpenny, and P. Sandborn, "Ontology-Based Knowledge Representation for Product Life Cycle Concepts and Obsolescence Forecasting," to be published *Journal of Computing and Information Science in Engineering*.

P. Sandborn, A. Konoza, R. Nelson III, D. Gerdes, and M. Shamet, "The Evaluation of End-of-Repair/End-of-Maintenance Dates for Electronic Assemblies," *Proceedings of the DMSMS Conference*, Hollywood, FL, August 2011.

References

[1] P. Sandborn, "Trapped on Technology's Edge," *IEEE Spectrum,* vol. 45, no. 4, pp. 42-58, April 2008.

[2] DoD 4140.1-R, "Supply Chain Materiel Management Regulation," 23 May 2003.

[3] C. Fine, Clockspeed: Winning Industry Control in the Age of Temporary Advantage, Reading, MA: Perseus Books, 1998.

[4] M. Pecht and S. Tiku, "Bogus: electronic manufacturing and consumers confront a rising tide of counterfeit electronics," *IEEE Spectrum,* vol. 43, no. 5, pp. 37- 46, May 2006.

[5] Y. Song and H. Lau, "A periodic-review inventory model with application to the continuous-review obsolescence problem," *European Journal of Operational Research,* vol. 159, no. 1, pp. 110-120, 16 November 2004.

[6] M. Drake, "Sharing obsolescence information at Raytheon with the component obsolescence and reuse tool," in *Reliability and Maintainability Symposium*, 2003.

[7] S. Wu, B. Aytac, R. Berger and C. Armbruster, "Managing Short Life-Cycle Technology Products for Agere Systems," *Interfaces,* vol. 36, no. 3, p. 234–247, May 2006.

[8] P. Sandborn, "Strategic Management of DMSMS in Systems," *Defense Standardization Program Journal,* pp. 24-30, April 2008.

[9] P. Sandborn and J. Myers, "Designing Engineering Systems for Sustainability," in *Handbook of Performability Engineering*, K. Misra, Ed., Springer London, 2008, pp. 81-103.

[10] Avnet, "The Best Defense is a Good Offense:Authorized Component Distributors Help Military/Aerospace OEMs Plan for Obsolescence," *White Paper,* 2007.

[11] W. Tomczykowski, "A study on component obsolescence mitigation strategies and their impact on R&M," in *Reliability and Maintainability Symposium*, 2003.

[12] J. Bogdanski, "Obsolescence management: Another perspective," *Military Embedded Systems,* vol. 5, no. 1, pp. 00-00, 2009.

[13] M. Pecht and D. Humphrey, "Addressing Obsolescence—The Uprating Option," *IEEE Transactions on Components and Packaging Technologies,* vol. 31, no. 3, pp. 741-745, September 2008.

[14] R. Stogdill, "Dealing with obsolete parts," *IEEE Design & Test of Computers,* vol. 16, no. 2, pp. 17-25, 1999.

[15] R. M. Robbins, "PROACTIVE COMPONENT OBSOLESCENCE MANAGEMENT," *A-B Journal,* vol. 10, no. 4, pp. 49-54, 2003.

[16] SiliconExpert, "Proactive vs. Reactive Approaches to Obsolescence Management," *White Paper,* 2009.

[17] T. J. Herald, "Technology refreshment strategy and plan for application in military systems a “How-to systems development process” and linkage with CAIV," in *National Aerospace and Electronics Conference (NAECON)*, 2000.

[18] A. L. Henke and D. Lai, "Automated Parts Obsolescence Prediction," in *DMSMS*

Conference, San Antonio, Texas, 1997.

[19] C. Josias, J. Terpenny and K. McLean, "Component Obsolescence Risk," in *2004 Industrial Engineering Research Conference (IERC)*, Houston, Texas, 2004.

[20] R. Solomon, P. Sandborn and M. Pecht, "Electronic part life cycle concepts and obsolescence forecasting," *IEEE Transactions on Components and Packaging Technologies,* vol. 23, no. 4, pp. 707-717, December 2000.

[21] M. J. Gravier and S. M. Swartz, "The dark side of innovation: Exploring obsolescence and supply chain evolution for sustainment-dominated systems," *The Journal of High Technology Management Research,* vol. 20, no. 2, pp. 87-102, 2009.

[22] P. Sandborn, F. Mauro and R. Knox, "A Data Mining Based Approach to Electronic Part Obsolescence Forecasting," *IEEE Transactions on Components and Packaging Technologies,* vol. 30, no. 3, pp. 397-401, September 2007.

[23] M. Meixell and S. Wu, "Scenario analysis of demand in a technology market using leading indicators," *IEEE Transactions on Semiconductor Manufacturing,* vol. 14, no. 1, pp. 65-75, February 2001.

[24] A. Meyer, L. Pretorius and J. Pretorius, "A model using an obsolescence mitigation timeline for managing component obsolescence of complex or long life systems," in *IEEE International Engineering Management Conference*, 2004.

[25] P. Singh and P. Sandborn, "Obsolescence Driven Design Refresh Planning For Sustainment-Dominated Systems," *Engineering Economist,* vol. 51, no. 2, pp. 115-139, 2006.

[26] G. Z. Porter, "An Economic Method for Evaluating," *White Paper,* pp. 1-9, May 1998.

[27] U. D. Kumar and H. Saranga, "Optimal selection of obsolescence mitigation strategies using a restless bandit model," *European Journal of Operational Research,* vol. 200, no. 1, pp. 170-180, 1 January 2010.

[28] L. Zheng, J. Terpenny, P. Sandborn and R. Nelson III, "Design Refresh Planning Models for Managing Obsolescence," in *ASME International Design Engineering Conferences & Computers and Information in Engineering Conference*, Chicago, 2012.

[29] T. Herald and J. Ramirez-Marquez, "System Element Obsolescence Replacement Optimization via Life Cycle Cost Forecasting," *IEEE Transactions on Components and Packaging Technologies,* (accepted for publication) 2012.

[30] T. Ōno, Toyota Production System: Beyond Large-scale Production, Cambridge, MA: Productivity, 1988.

[31] S. Nair and W. Hopp, "A model for equipment replacement due to technological obsolescence," *European Journal of Operational Research,* vol. 63, no. 2, pp. 207-221, 10 December 1992.

[32] K. D. Cattani and G. C. Souza, "Good buy? Delaying end-of-life purchases," *European Journal of Operational Research,* vol. 146, no. 1, pp. 216-22, April 2003.

[33] R. H. Wilson, "A Scientific Routine for Stock Control," *Harvard Business Review,* vol. 13, no. 1, pp. 116-128, October 1934.

[34] Y. Zenou, *Discounting: Discrete versus Continuous Compounding,* Stockholm: Research Institute of Industrial Economics, 2006.

[35] P. Sandborn, Cost Analysis of Electronic Systems, Singapour: World Scientific, 2012.

[36] F. W. Harris, "How Many Parts To Make At Once," *Factory: The Magazine of Management,* vol. 10, no. 2, pp. 135-136, February 1913.

[37] D. Panda, S. Kar and M. Maiti, "Multi-item EOQ model with hybrid cost parameters under fuzzy/fuzzy-stochastic resource constraints: A geometric programming approach," *Computers & Mathematics with Applications,* vol. 56, no. 11, pp. 2970-2985, 2008.

[38] Y. Chiou and L. Lan, "Ordering and Warehousing Strategies for Multi-Item Multi-Branch Firm's Inventory: Clustering Approaches," *Journal of the Eastern Asia Society for Transportation Studies,* vol. 6, pp. 2809 - 2821, 2005.

[39] K. Kotb and H. Fergany, "Multi-Item EOQ Model with Both Demand-Dependent Unit Cost and Varying Leading Time via Geometric Programming," *Applied Mathematics,* vol. 2, no. 5, pp. 551-555, 2011.

[40] M. Bhagoria, C. Sadiwala and V. Khare, "Economic order quantity for multiple items in resource constraints," *Indian Journal of Science and Technology,* vol. 3, no. 6, pp. 707-709, 2010.

[41] K. Kotb and H. Fergany, "Multi-Item EOQ Model with Varying Holding Cost: A Geometric Programming Approach," *International Mathematical Forum,* vol. 6, no. 23, pp. 1135 - 1144, 2011.

[42] S. Mondal and M. Maiti, "Multi-item fuzzy EOQ models using genetic algorithm," *Journal of Computers and Industrial Engineering,* vol. 44, no. 1, pp. 105-117, 2003.

[43] J. S. Arora, Introduction to optimum design, San Diego, CA: Elsevier Academic Press, 2004.

[44] R. Nelson III, P. Sandborn, J. P. Terpenny and L. Zheng, "Modeling Constraints in Design Refresh Planning," in *Proceedings of the ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2011*, Washington, DC, 2011.

[45] G. Dantzig, Linear programming and extensions, Princeton, NJ: Princeton University Press, 1963.

[46] R. Courant, "Variational methods for the solution of problems of equilibrium and vibrations," *Bull. Amer. Math. Soc.,* vol. 49, pp. 1-23, 1943.

[47] AIR FORCE LOGISTICS MANAGEMENT AGENCY GUNTER AFB AL , "Air Force Bad Actor Program," 1997.

[48] United Nations Framework Convention on Climate Change, "Kyoto Protocol Reference Manual, On Accounting of Emissions and Assigned Amount," 2008.

[49] P. Sandborn, V. Prabhakar and O. Ahmad, "Forecasting electronic part procurement lifetimes to enable the management of DMSMS obsolescence," *Microelectronics Reliability,* vol. 51, no. 2, pp. 392-399, February 2011.

[50] Electronic Industries Alliance, *Product Life Cycle Data Model (ANSI/EIA-724-97),* 1997.

[51] "SiliconExpert Technologies," SiliconExpert Technologies, Inc., [Online]. Available: http://www.siliconexpert.com/. [Accessed 21 11 2011].

[52] "Total Parts Plus," Total Parts Plus, Inc., [Online]. Available: http://home.totalpartsplus.com/. [Accessed 21 November 2011].

[53] "Information Handling Services (IHS)," IHS Inc., [Online]. Available:

http://www.ihs.com/. [Accessed 21 November 2011].

[54] "QStar Technologies," QStar Technologies, [Online]. Available: http://www.qstar.com/. [Accessed 21 November 2011].

[55] "PartMiner Worldwide," PartMiner Inc., [Online]. Available: http://www.partminer.com/main/. [Accessed 21 November 2011].

[56] S. Lee and I. Grossmann, "New algorithms for nonlinear generalized disjunctive," *Computers & Chemical Engineering,* vol. 24, no. 9/10, p. 2125–2141, 2000.

[57] A. Charnes and W. Cooper, "CHANCE-CONSTRAINED PROGRAMMING," *Management Science,* vol. 6, no. 1, pp. 73-79, October 1959.

[58] J. R. Birge and F. Louveaux, Introduction to Stochastic Programming, New York: Springer, 1997.

[59] R. A. Morris, P. Morris, L. Khatib and N. Yorke-Smith, "Temporal Planning with Preferences and Probabilities," in *Proceedings of ICAPS'05 Workshop on Constraint Programming for Planning and Scheduling*, Monterey, CA, 2005.

[60] S. S. Rao, Engineering Optimization: Theory and Practice, 4 ed., Hoboken, NJ: John Wiley & Sons, 2009.

[61] E. Board, "Mathematical Programming Glossary," INFORMS Computing Society, 2010. [Online]. Available: http://glossary.computing.society.informs.org/index.php?page=E.html. [Accessed 25 July 2012].

[62] EUROPEAN PARLIAMENT AND OF THE COUNCIL, "EUROPEAN PARLIAMENT AND OF THE COUNCIL," *DIRECTIVE 2002/95/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL,* pp. 19-23, 2003.

[63] P. Sandborn and R. Nelson III, "Constraint-Driven Refresh Planning of Systems Subject to Obsolescence," in *DMSMS Conference*, Palm Springs, CA, 2008.

[64] J. Myers and P. Sandborn, "Integration of Technology Roadmapping Information and Business Case Development into DMSMS-Driven Design Refresh Planning of the V-22 Advanced Mission Computer," in *Aging Aircraft Conference*, Palm Springs, California, 2007.

[65] P. Singh, "FORECASTING TECHNOLOGY INSERTION CONCURRENT WITH DESIGN REFRESH PLANNING FOR COTS-BASED OBSOLESCENCE SENSITIVE SUSTAINMENT-DOMINATED SYSTEMS," College Park, MD, 2004.

[66] P. Spiegel, "AIR FORCE OBLIGATED TO KEEP AGING PLANES ; CONGRESS ENACTS," *The Sun,* p. 10A, 27 August 2006.

[67] J. McGee, "Enterprize Systems Journal," 11 October 2005. [Online]. Available: http://esj.com/Articles/2005/10/11/Legacy-Systems-Why-History-Matters.aspx?p=1. [Accessed 21 November 2011].

[68] J. Bisbal, D. Lawless, B. Wu and J. Grimson, "Legacy information systems: issues and directions," *IEEE Software,* vol. 16, no. 5, pp. 103-111, September 1999.

[69] Continuing Consolidation of the Statutes of Manitoba, *Green Energy Equipment Tax Credit Regulation (C.C.S.M. c. I10),* 2008.

[70] California Alternative Energy and Advanced Transportation Financing Authority, *Senate Bill No. 71,* 2010.

[71] United States Code, *TITLE 26 > Subtitle A > CHAPTER 1 > Subchapter A > PART IV > Subpart E > § 48. ENERGY CREDIT,* 2010.

[72] A. De Grip and J. Van Loo, "The economics of skills obsolescence: A review," *Research in Labor Economics,* vol. 21, pp. 1-26, 2002.

[73] W. Arthur Jr., W. Bennett Jr., P. L. Stanush and T. L. McNelly, "Factors That Influence Skill Decay and Retention: A," *Human Performance,* vol. 11, no. 1, pp. 57-101, March 1998.

[74] C. Grinstead and J. Snell, Introduction to probability, 2 ed., Providence, RI: American Mathematical Society, 1997.

[75] S. Kotz, Y. Lumelskii and M. Pensky, The stress-strength model and its generalizations: theory and applications : P(X<Y), Singapore: World Scientific, 2003.

[76] I. Elishakoff, Safety factors and reliability : friends or foes?, Boston: Kluwer Academic Publishers, 2004.