Proceedings of the ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2011
August 29-31, 2011, Washington, DC, USA

DETC2011-47644

MODELING CONSTRAINTS IN DESIGN REFRESH PLANNING

Raymond Nelson III, Peter Sandborn
CALCE Center for Advanced Life Cycle Engineering
Department of Mechanical Engineering
University of Maryland
College Park, MD 20742, USA

Janis P. Terpenny and Liyu Zheng
Center for e-Design
Virginia Polytechnic Institute & State University
Blacksburg, VA 24061, USA

ABSTRACT

When an original equipment manufacturer no longer supplies and/or supports a product then the product is considered to be obsolete. Obsolescence is a significant problem for systems whose operational and support life is much longer than the procurement lifetimes of their constituent components. Unlike high-volume, commercial products, which are quickly evolved, long field life, low-volume systems, such as aircraft may require updates of their components and technology called design refreshes to simply remain manufacturable and supportable. However these systems can't perform design refreshes all the time due to the high non-recurring and re-qualification costs. One approach to optimally managing this problem is to use DRP (Design Refresh Planning), which is a strategic method for scheduling design refreshes such that the life cycle cost impact of obsolescence is minimized. The planning of these design refreshes is restricted by various constraints, which need to be implemented into the DRP process. These constraints can reflect technology roadmap requirements, obsolescence management realities, logistical restrictions, budget ceilings and management policy. In this paper, constraints imposed on the DRP process are explored, classified within a taxonomy, and implemented in the planning process. A communications system design example is included.

Keywords: design refresh planning, obsolescence, DMSMS, product life-cycle management (PLM), constraints

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 paper 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. In this paper, "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 clock speed products [2], 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 [3].

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 [4]. Sustainment in this paper 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. Examples of

sustainment-dominated systems include airplanes, power plant controls, medical systems, military systems, 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 [5]. 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. 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.

Effective long-term management of DMSMS in systems requires addressing the problem on three different management levels: reactive, pro-active and strategic.

Reactive management level is concerned with determining an appropriate, immediate resolution to the problem of components that are obsolete or soon will be. Common reactive DMSMS management approaches include: lifetime buy, bridge buy, alternative or substitute parts, buying from aftermarket sources, uprating, emulation, and salvage [6]. 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.

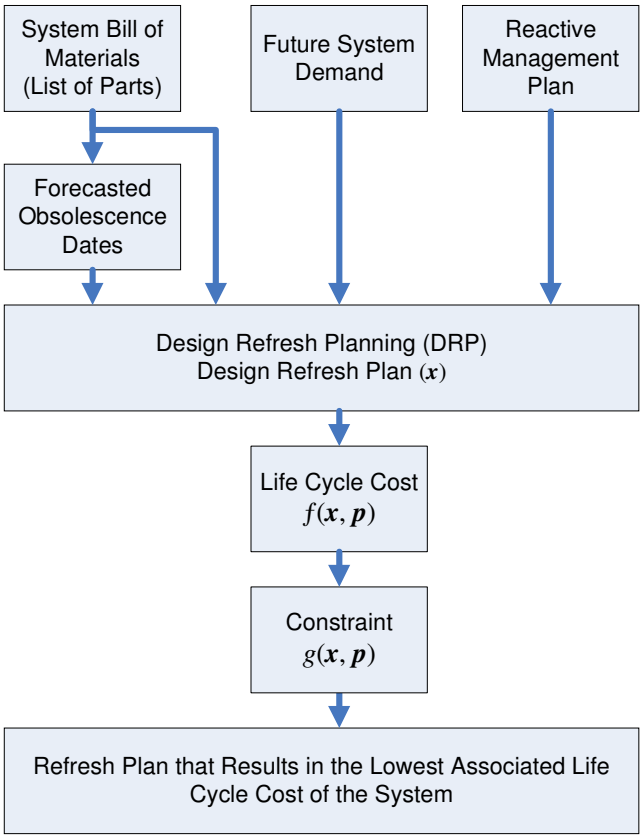

Figure 1. DESIGN REFRESH PLANNING (DRP) PROCESS SHOWING THE REQUIRED INPUT DATA AND THE RESULTING OUTPUT.

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.

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. 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.

Section 2 describes the design refresh planning process and Sections 3 and 4 focus on constraint formation and implementation.

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.

Figure 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[1] for the components in the bill of materials, the future demand for the system being produced and sustained

[1] Forecasting the date on which original manufacturers of electronic components discontinue the components (the obsolescence date) has been previously treated in the literature, e.g., [7-9] and is commercially available from several sources. Availability of components from the aftermarket may extend the effective obsolescence date for some components, however, not every component is available in the aftermarket and depending on the nature of the system being supported, aftermarket parts may not be an acceptable solution due to counterfeit and other risks.

(future manufacturing needs and spare parts required to maintain fielded systems), and the reactive management plan (applied between refreshes).

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 is obsolescence dates for all the components in the BOM. The DRP process simulates a timeline of events based on the input data and generates various combinations of design refresh dates. 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. Once associated with a cost, a series of constraints are applied to the plans (to identify the plans that are not feasible). The design refresh plan that has the lowest associated life cycle cost out of the selection of feasible plans is then selected.

The DRP problem can be formulated as shown in Eqn. (1):

$$\text{minimize:} \quad f(\boldsymbol{x},\boldsymbol{p}) = \sum_{i=1}^{n} \frac{Q_i C_i}{\left(1+\frac{R}{100}\right)^{d_i}} + \sum_{j=1}^{r} \frac{NRE_j}{\left(1+\frac{R}{100}\right)^{d_j}} \quad (1)$$

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

where,

Q_i	Quantity of systems to be manufactured at the *i*th manufacturing event, including spares
C_i	Recurring cost of manufacturing a system instance at the *i*th manufacturing event, including spares
NRE_j	Non-recurring cost of the *j*th design refresh
n	Number of manufacturing events
r	Number of design refreshes in the plan
R	After tax discount rate on money
d_i ; d_j	Difference in years between *i*th/*j*th manufacturing/design refresh event date and the base year for money
k	Index used to identify constraint
K	Number of constraints
m	Number of parameters

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

For parameters, it is assumed that they cannot be changed (i.e., they have uncontrollable variations within a known range). 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 values stored as parameters can be the production schedule, forecast obsolescence dates, and costs for different DRP activities. Since the values used in the design variable vector and the set of parameters represent monetary and quantitative amounts, $\boldsymbol{x}$ and $\boldsymbol{p}$ are restricted to real values.

The next section will discuss the types of design refresh planning constraints that can be imposed.

CONSTRAINT TAXONOMY

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

The constraint taxonomy proposed in this paper was created from the viewpoint of an organization sustaining a system. This organization can be a private company, government agency, or any group sustaining a system (e.g., desktop computers or aircraft). The term sustaining has the same meaning as earlier defined, which succinctly means to maintain existing systems and manufacture new systems. Relativistic words such as "I" or "internal" refer to the organization sustaining the system.

Figure 2 shows a constraint taxonomy that classifies constraints based on the permission and ability to perform a design refresh activity (the taxonomy in Fig. 2 is akin to the classic English lesson that differentiates "May I" and "Can I").

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

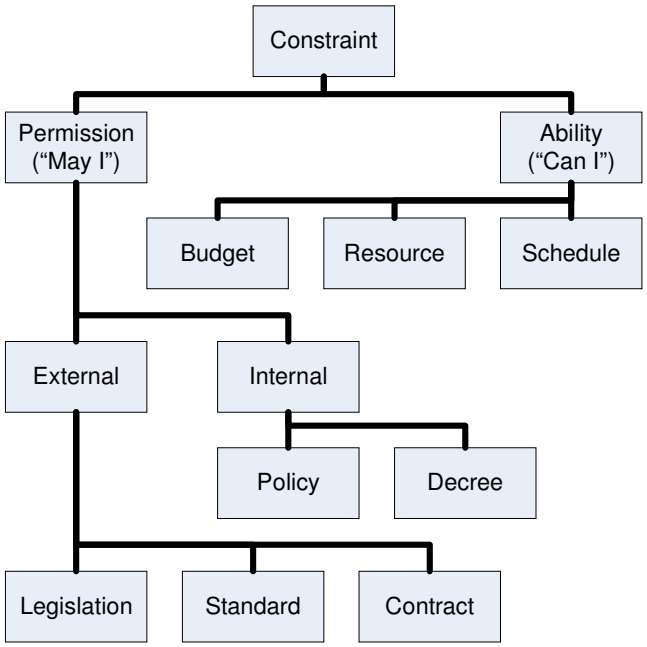

Figure 2. CONSTRAINT TAXONOMY.

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 customer who purchases the systems that the organization manufactures. Any contractual commitments between the two that affect the design refresh activities performed by the organization sustaining a system creates permission-based constraints. 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, maximum or optimum) for quality or level of performance. Any standards adhered to by an organization in order to establish a certification of some type 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 a 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 specific aspects of the life cycle planning process and are generally more limited 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 from performing a design refresh activity, rather the actors within the organization who violate these permission-based constraints will suffer various disciplinarian 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. 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: schedule, budget and resource. 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 Fig. 2, information on the aspects of the system restricted by the constraints needs to be determined.

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 constrains 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 limit 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 can only be performed at its home base).

This paper focuses on how constraints restrict temporally and specifically how they are implemented in the management of sustainment-dominated electronic systems. Constraints that constrain logistically and financially in general are less difficult to implement, whereas constraints that constrain temporally require more derivation and formulation. For the DMSMS affected systems, constraints that restrict temporally are the most prevalent DRP drivers.

Temporally Restrictive Constraints

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 such as obsolescence events, legislation enactment events, changes in standards events, etc., because a large majority of system constraints are geared toward mitigating the impacts of those events.

Constraints that constrain temporally are usually bounded ranges of time that require one or more design refresh activities to complete within them; however, those 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 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.

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 through 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.

"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 paper) 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.

"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).

A recent example of a system constraint that resulted in an organization identifying a component's obsolescence event as "Strong A" was caused by the European legislation called the Restriction of Hazardous Substances (RoHS) Directive [10]. This legislation 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.

"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[2] and any new instances of the system will have to be delivered with a non-obsolete operating system.

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.

Constraint Implementation

Component instances that are in a "Strong" obsolescence event category will be examined because only the "Strong"

[2] 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 determined by reviewing all fielded versions of the system and accumulating appropriate quantities of affected system elements, see Constraint Implementation.

obsolescence events result in modification of the DRP process. 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 dates, can include uncertainties in this algorithm. This is important especially since the input uncertainty is often large for DRP problems.

Step 1: Create the Constraint. 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 (C_S) and the constraint end date (C_E), which form the constraint period. The constraint start date is calculated by subtracting the look-ahead time (*LAT*) from the forecasted date of obsolescence (D_o). *LAT* 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.

$$C_S = D_o - LAT \tag{2}$$

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

Equation (2) determines the constraint start date (C_S); however, it needs to be expressed as an inequality constraint in order to be applied to the life cycle cost minimization problem in Eqn (1). The first of a pair of explicit constraints can be written as:

$$g_1(\boldsymbol{x}) = \boldsymbol{x} - C_S \leq 0 \tag{3}$$

It should be noted that $\boldsymbol{x}$ in Eqn. (3) is the design variable, which in this case is a design refresh plan with r number of design refresh dates placed in a vector. Any design refresh date can satisfy the constraint and rather than write out r constraints with the individual design refresh dates, which would be cumbersome, the constraint is left in this form with the understanding that $\boldsymbol{x}$ is a vector of values, each of which can satisfy the constraint.

The constraint end date (C_E) 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 (D_P) 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 C_S and C_E consists of two periods: the look-ahead time and the "waiting time" (*WT*). 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. The look-ahead time and waiting time durations are defined by Eqn. (2) and (4) respectively,

$$WT = C_E - D_O = D_P - D_O \tag{4}$$

The "waiting time" was defined to help distinguish between the constraints created from the "Strong A" and "Strong B" obsolescence types. 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 following production event.

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 Eqn. (2) can be used to find the constraint start date (C_S). 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 (C_E) is the same as the obsolescence date (D_o), (i.e., $C_E = D_o$). 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. With the constraint end date known, the second of a pair of explicit constraints can be written as:

$$g_2(\boldsymbol{x}) = C_E - \boldsymbol{x} \leq 0 \tag{5}$$

The combined pair of Eqn. (3) and (5) forms an overlap of inequality constraints that result in a bounded range. This range is the constraint period during which a design refresh (x) must complete its activities.

The cost of the backfit process can be broken up 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 5 in Case Study). 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, except when it comes to creating "Strong A" constraints. This inserted production event should not be used as a constraint end point since by definition the "Strong A" component can remain a constituent component 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.

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 sustainment-dominated systems, therefore, a component's replacement (at a design refresh) may also go obsolete before the end of the system's operational life. In order to model this effect, the components that replace the original component 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: simulates a replacement for the predecessor component[3] that went obsolete; generates the replacement's obsolescence date; and if the replacement's obsolescence event is before the end of support date of the system, simulates 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 [9], 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 [11]. The life cycle curve is divided into regions that reflect the rate of a component's maturity that correspond to the following life cycle codes: 1 = emerging, 2 = growth, 3 = maturity, 4 = decline, 5 = phase out, 6 = obsolete. 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 (6) is used to generate the obsolescence date of new components introduced at design refreshes.

$$D_o = D_{pc} + L\left[\frac{I_o - I_R}{I_o - I_I}\right] \tag{6}$$

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 components can also be obtained from commercial electronic component databases.

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 constraint will determine how the calculated obsolescence date is used.

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). A production event must follow the

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

obsolescence date for the synthesized replacement component otherwise no constraint is required.

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.

Step 3: Constraint Implementation. In general, all constraints are applied to the design variable and joined with a logical "AND" so that all constraints must be satisfied for a design refresh plan to be considered viable; however, the temporal constraints developed in the algorithm described in this paper are applied in a different way. The temporal constraints developed for the DRP process are grouped into pairs (i.e., bracketed) because they share a common variable. Each half of a constraint pair bounds a positive or negative infinite range, which when the constraints are joined with a logical "AND", together they bound a limited range. For a unique constraint pair, only one value of the design variable vector is needed to satisfy the unique constraint pair.

As described in the previous steps in this algorithm, these constraints are dependent on the obsolescence dates and production dates. Because of the input uncertainty inherent in these dates, it is possible and quite common that no single unique refresh plan will be viable for all possible variations of the system and constraints. Therefore, the constraints are applied after all the candidate refresh plans are generated rather than generating a subset of plans (e.g., via design-space searching or an iterative method). Post processing the solution with constraints ensures that all possible combinations of refresh plans are evaluated on an equal basis. For example, if the input uncertainty is modeled using probability distributions for all input data using a Monte Carlo approach, the same sampled input data used to calculate the life cycle cost for associated design refresh plans is the same sampled data used to generate all constraints. With uncertainty modeled identically in both the DRP process and the constraint generating method, it is possible to assess the probability that individual design refresh plans will satisfy all the constraints.

MODELING UNCERTAINTY

DRP models that incorporate input uncertainty have been developed [12]. However, existing DRP models do not assess the probability of a design refresh plan not satisfying all constraints nor do they estimate the cost of design refresh plans that violate constraints.

In order to determine the probability that the design refresh plan will satisfy all imposed constraints, it is necessary to determine the probability of each constraint being satisfied. The product of all the probabilities equals the probability that a design refresh plan will satisfy all constraints.

For example, consider the special case where the time a design refresh "looks-ahead" into the future at the beginning of the design refresh 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 constraint that requires a design refresh to resolve the obsolescence of a component and complete its activities before the next production event is imposed. To determine the probability that a design refresh will satisfy the imposed constraint, which requires it to complete its activities between the obsolescence event of the component and the following production event is done by creating a probability aggregate, which is the product of the probabilities that the order of events is logical. For example, the date a component goes obsolete ($d_{Obsolescence}$) must take place before a design refresh ($d_{Refresh}$) that resolves the problems the obsolescence event created completes its activities that should then be followed by the production date ($d_{Production}$) that incorporates the updates made by the design refresh. This is done by taking the probability for the correct chronological order of events and multiplying them together to form a probability aggregate:

$$P_1 = P\left(d_{Obsolescence} < d_{Refresh}\right) P\left(d_{Refresh} < d_{Production}\right) P\left(d_{Obsolescence} < d_{Production}\right) \quad (7)$$

Equation (7) is gives the probability for one design refresh satisfying one constraint. If there are multiple constraints that need to be satisfied then probabilities for the remaining constraints being satisfied need to be determined to compute the final probability of a design refresh plan satisfying all constraints.

A less efficient but more versatile method for approximating the probability that a design refresh plan satisfies the constraints is to use a Monte Carlo method and simply run the DRP model a statistically significant number of times, and determined the fraction of runs in which the refresh plan satisfied all constraints.

CASE STUDY

To demonstrate the design refresh planning process with constraints, a case study was performed based on a portion of a 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 this example includes supporting as well as producing several instances of the server cabinet design. Table 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 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

For this case study, the DRP modeling environment used is a DRP software tool called MOCA (Mitigation of Obsolescence Cost Analysis) [12], 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 cost 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.

The first system constraint for this case study is the Restriction of Hazardous Substances Directive (RoHS). This constraint identifies any electronic component that is not RoHS compliant as the components being restricted from being built into new systems beyond a specified date. Because this constraint only prohibits the manufacturing of new systems containing these components, then this system constraint has identified the date at which non-compliant electronic components are prohibited from being built into new systems as a "Strong A". There were 7 components identified as causing a "Strong A" obsolescence event.

The second system constraint 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 constraint applies to all software used in the system. Because it affects the operation of the components it also affects the manufacturing of systems containing those components, thus this system constraint identifies the obsolescence event of these components as "Strong B". With the component types identified as causing a "Strong B" obsolescence event, the system whose BOM is composed of 79 total components is searched for the affected component types. There were 12 components identified with the component types restricted by the policy.

For the sake of brevity, Tab. 2 provides obsolescence information[4] on the three selected example components.

The constraint-generating algorithm is repeated to ensure all constraints for the Driver, Hardware, and Software are formed. The resulting pairs of explicit temporal constraints are shown in Tab. 3.

Table 2. SUBSET OF COMPONENTS WHOSE CONSTRAINT SYNTHESIS WILL BE DEMONSTRATED IN THIS CASE STUDY.

Component	Obsolescence Date	Procurement Lifetime (years)	RoHS Complient
Driver	2007.5	10	N/A
Hardware	2018	30	No
Software	2009.54	10	N/A

Table 3. CONSTRAINTS FOR CASE STUDY.

Component	Start Date (C_S) (See Eqn. (3))	End Date (C_E) (See Eqn. (5))
Driver	$g_1(x)=x-2004.50\leq 0$	$g_2(x)=2008.00-x\leq 0$
Hardware	$g_3(x)=x-2011.00\leq 0$	$g_4(x)=2015.00-x\leq 0$
Software	$g_5(x)=x-2006.54\leq 0$	$g_6(x)=2009.54-x\leq 0$
Software	$g_7(x)=x-2015.54\leq 0$	$g_8(x)=2017.54-x\leq 0$

The assumptions for the DRP process are: look-ahead time (*LAT*) is set to 3 years, RoHS compliance date of January 1, 2014 an end of support date 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.

Figure 3 shows the results of the MOCA analysis of the example described in this section without applying constraints and without uncertainty.[5] Figure 4 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 graphs 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 vertical axis shows the corresponding total life cycle cost. The data points each represent unique design refresh plans (unique combinations of design refresh dates). The shape of the data point indicates how many design refreshes are in the refresh plan. The filled circles are plans that consist of exactly one refresh (i.e., one design refresh date); the triangles have exactly two refreshes in their plans; the square represents plans with exactly 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

[4] 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.

[5] 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 [12].

refreshes). The points in Fig. 3 and 4 are plotted at the average dates of the refreshes in the plan.

Figures 3 and 4 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 4 shows that many refresh design plans can be created; however, once temporal constraints that reflect system constraints 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 Fig. 4. For comparison, the best refresh plan circled in Fig. 4 is also circled in both Fig. 3 and Fig. 5. 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 Tab. 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 a symmetric distribution) – normal distributions were chosen for convenience. After running 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 Fig. 5, 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 Fig. 5 and 6 the vertical-axis is the mean life cycle cost. Figure 6 shows the nine refresh plans from Fig. 5 with a 10% or less probability of failure. Note, no plans have 0% probability of failure (i.e., 100% probability of satisfying all the constraints when uncertainties are considered). Since both life cycle cost and design refresh plan probability of failure should be minimized, this becomes a multi-objective problem where the grouping of refresh plans creates a Pareto frontier.

A weighted sum method can determine a best solution; however, the life cycle cost can be related to design refresh plan probability of failure to make a single objective.

If the cost of violating a constraint could be determined, then it would allow an expected cost value for each plan to be

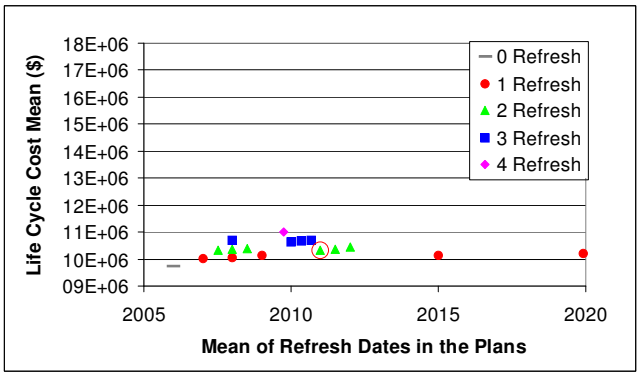

Figure 3. MOCA GENERATED REFRESH PLAN WITH NO CONSTRAINTS APPLIED.

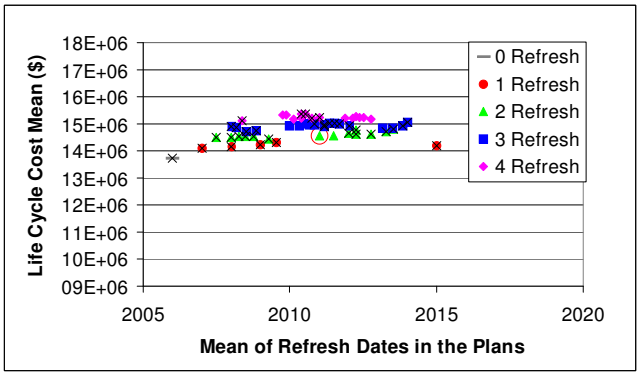

Figure 4. MOCA GENERATED REFRESH PLAN WITH A DETERMINISTIC APPLICATION OF THE GENERATED CONSTRAINTS. THE CROSSED-OUT POINTS DO NOT SATISY ONE OR MORE OF THE CONSTRAINTS.

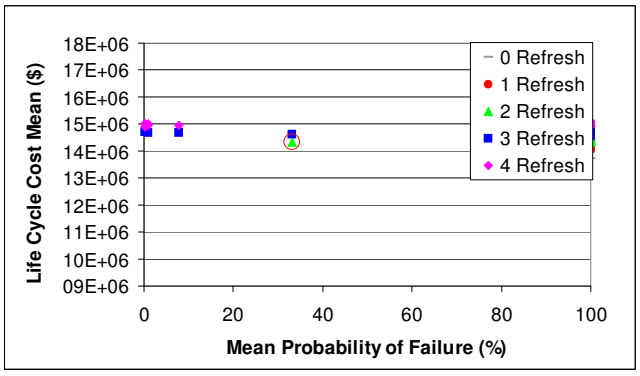

Figure 5. MOCA GENERATED REFRESH PLANS WITH THE APPLICATION OF STATISTICAL PARAMETERS ACCOUNTING FOR UNCERTAINTY.

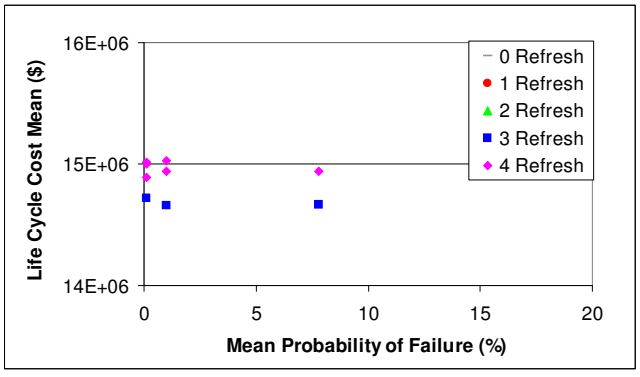

Figure 6. MOCA GENERATED REFRESH PLANS WITH APPLICATION OF STATISTICAL PARAMETERS ACCOUNTING FOR UNCERTAINTY.

calculated, which would allow the minimization of the expected life cycle cost. The design refresh plan with the lowest expected life cycle value would be the best plan.

Uncertainty 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.

DISCUSSION

This paper describes the obsolescence problem for sustainment-dominated systems and focuses on modeling constraints within the DRP methodology. A constraint taxonomy is presented that is used to classify constraints based on whether the constraint is based on an organization (that is sustaining a system) has the permission or ability to perform a design refresh.

A detailed treatment of temporal constraints was presented including an algorithm used to generate temporal constraints using obsolescence event type definitions to identify what sustainment activities are affected by a component's obsolescence event.

Finally, a case study was presented to demonstrate the effect constraints have on the best solution to a design refresh planning problem. The main idea presented in this paper is that the addition of constraints to the DRP methodology is necessary and its effect on the results is significant.

Future work will be to refine the input uncertainty portion of the DRP methodology to include penalty costs due to various scheduling infractions such as a design refresh completing later than scheduled.

ACKNOWLEDGEMENT

The authors acknowledge the National Science Foundation (Division of Design and Manufacturing Innovation) Grant Nos. CMMI 928628, 928837 and 928530 for their support. The authors 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.

REFERENCES

[1] Sandborn, P., 2008. "Trapped on technology's trailing edge". *IEEE Spectrum*, **45**(1) pp. 42-45.

[2] Fine, C., 1998. *Clockspeed: Winning industry control in the age of temporary advantage*. Perseus Books, Reading, MA.

[3] Pecht, M., and Tiku, S., 2006. "Bogus! Electronic manufacturing and consumers confront a rising tide of counterfeit electronics". *IEEE Spectrum* **43** pp. 37-46.

[4] Sandborn, P., and Myers, J., 2008. "Designing Engineering Systems for Sustainment". In *Handbook of Performability Engineering*, K. Misra, ed., Springer, London, pp. 81-103.

[5] Tomczykowski, W., 2003. A study on component obsolescence mitigation strategies and their impact on R&M, *Proceedings Series*, Annual Reliability and Maintainability Symposium (RAMS), Tampa, FL, pp. 332-338.

[6] Stogdill, R., 1999. "Dealing with obsolete parts". *IEEE Design and Test of Computers*, **16** pp. 17-25.

[7] Solomon, R., Sandborn, P., and Pecht, M., 2000. "Electronic part life cycle concepts and obsolescence forecasting". *IEEE Transactions on Components and Packaging Technologies* **23** pp. 707-713.

[8] Josias, C., and Terpenny, J., 2004. Component obsolescence risk assessment, *Proceedings Series*, Industrial Engineering Research Conference (IERC).

[9] Sandborn, P., Prabhakar, V., and Ahmad, O., 2011. "Forecasting technology procurement lifetimes for use in managing DMSMS obsolescence". *Microelectronics Reliability* **51** pp. 392-399.

[10] Directive 2002/95/EC of the European Parliament and of the Council, of 27 January 2003.

[11] ANSI/EIA-724. 1997. Product Life Cycle Data Model.

[12] Singh, P., and Sandborn, P., 2006. "Obsolescence driven design refresh planning for sustainment-dominated systems". *The Engineering Economist* **51** pp. 115-139.