2018 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
SYSTEMS ENGINEERING (SE) TECHNICAL SESSION
AUGUST 7-9, 2018 - NOVI, MICHIGAN

SET-BASED DESIGN AND OPTIMIZATION.... CAN THEY LIVE TOGETHER AND MAKE BETTER TRADE SPACE DECISIONS?

Stephen Rapp, PhD
Tank Automotive Research Development Engineering Center (TARDEC)
Warren, MI

Norbert Doerry, PhD
Naval Sea Systems Command
Washington, DC

Ratna Chinnam, PhD
Leslie Monplaisir, PhD
Alper Murat, PhD
Gary Witus, PhD
Industrial & Systems Engineering Department
Wayne State University
Detroit, MI

ABSTRACT

Often during Product Development, externalities or requirements change, forcing design change. This uncertainty adversely affects program outcome, adding to development time and cost, production cost, and can compromise system performance. We present a development approach that minimizes impacts, by proactively considering the possibility of changes in the externalities and mid-course design changes. The approach considers the set of alternative designs and the burdens of a mid-course change from one design to another in determining the relative value of a specific design through the set-based design methodology. The approach considers and plans parallel (redundant) development of alternative designs with progressive selection of options, including time-versus-cost tradeoffs and the impact change-costs. The approach includes a framework of the development process addressing design and integration lead-times, their relationship to the time-order of design decisions, and the time-dependent burden of design changes. We also compare set-based and single point design schemes.

INTRODUCTION

Product Development (PD) remains uncertain and fraught with risk in development as outside factors change over time. Changing requirements during PD play havoc with program budgets, resources and schedule. Stakeholders with different concerns, constraints and changeable priorities coupled with cost and engineering technical fulfilment uncertainty impact system choices through design and development in PD programs. A Government Accounting Office (GAO) audit of Navy Destroyer Programs addressed this issue

(Anti-Air Warfare requirements in this case) and proposed the need for expanded design space in future programs [1]. Set-Based Design (SBD) as a candidate engineering development approach, holds promise to add resiliency into the PD process by expanding the design space to consider set design solutions. SBD considers a set of design solutions and does not move to a point solution until PD uncertainty is resolved.

SBD is based on satisficing given multiple constraints (requirements). <Note: Satisficing is a decision-making technique that entails searching alternatives until the threshold is met.> Different stakeholders source different constraints and requirement thresholds change over time. The relative priorities, i.e., the willingness of stakeholders to relax one requirement to be able to restrict another requirement and still have feasible solutions, also change over time. SBD is an iterative process where design and requirements evolve in parallel and stakeholders restrict and relax requirements regarding feasible design space solutions [2]. It is a concurrent engineering process that helps stakeholders understand requirement interdependencies and impact on design as they work to develop the performance specification and preliminary design.

LITERATURE REVIEW INSIGHTS

SBD lacks a rigorous mathematical, quantitative formalization. Emergence of computational, combinatorial design generation and evaluation tools and methods continue, but there are limitations in their use and application as they mainly focus on "point solutions" even with pareto multi-criteria design. Optimal point solutions early in the design, are faced with unknowns that affect design fulfilment and success. Point designs are "brittle" as they must react to technical, time and cost changes. Ongoing requirements changes, in thresholds and priorities, also force point solution modification when expectations fall short of target. SBD however, creates set solutions, where set contains multiple point solutions in a region of the design space itself. These set solutions offer the potential to improve design resilience since set solutions increase confidence that a single design from the set will better meet requirements and are more tolerant to parameter changes over time. Thus, a solution contained in a region, compared to a point solution, and also being allowed to change over time, increases PD process resilience.

The general body of knowledge supports the theoretical foundation that uncertainty and risk severely impact designs as they change in the PD cycle. Multiple foundational examples of failed programs, particularly complex and high cost programs, are directly linked to a lack of rich alternatives in a robust design space. Furthermore, failures are tied to point design process brittleness, which is unable to deal with subsystem, assembly or component design failures associated with uncertainty over time in PD. SBD can expand and contract its set solution, in the face of uncertainty. There are multiple, successful SBD qualitative examples of this process resiliency. SBD is also directly linked to PD programs that are effectively using it to remove design process brittleness. However, the literature points to the need of coupling quantitative mathematical analysis into the PD process to create not just more process resiliency, but cost-effective, adaptive and technically improved solutions. The need for a framework that couples SBD with mathematical analytics is both real and overdue. Additionally, the concept to optimize, while keeping solution variance in concept sets has been proposed.

SBD is a proven qualitative process associated with organizational conference room decision making. This is not disparaging, but rather it is encouraging, in that SBD has proven itself in decision making. Therefore, designing a quantitative form of SBD that keeps the richness of SBD facing and agilely dealing with uncertainty in PD, is unique and novel. Furthermore, the novel approach and framework presented in this paper, goes beyond using uncertainty to expand design

space around point solutions, to actually creating true set solutions with their total uncertainty [3].

PROBLEM OVERVIEW

Many systems, especially military systems, have protracted development lifecycles. During development, various external factors that influence design decisions often change. The challenge is to develop a system that ends up being cost-effective and is cost effective to develop, despite changes in the externalities during development. A development process that meets these challenges is resilient with respect to changes in the externalities.

The externalities we are considering are: (1) the relative values of system performance, system burden, and unit production cost, and (2) the development cost, time and uncertainties of candidate technologies/ options. The external factors have a known value at any point in time, but their final value, when the development is over and the system enters production, are unknown until the end. They are indeed "random" variables. Traditional point design treats the externalities as "deterministic". As a result, reactions to changes can incur greater costs and/or performance compromises than if the development program had considered potential variability of the externalities.

The approach we are pursuing is Set-Based Design (SBD). SBD carries a (redundant) set of options and alternatives forward during development, incrementally winnowing and adjusting the set as development proceeds and in response to changes in the externalities, leading to a final point design for production. The principle of SBD is to keep a broad range of options under consideration "as long as possible" to provide resiliency to changes in the externalities.

FRAMEWORK FORMULATION

The high-level framework process uses a stochastic process to define the value, i.e. Contribution-to-Design, of developing multiple options, in a set from PD milestone/epoch to epoch. Contribution-to-Design is a combination of system performance, production cost, development time and cost. We treat the Contribution-to-Design as a black box treatment (allowing flexibility of application) to seed the values required to develop a Markov Decision Process. This is standard stochastic automata with utilities that presume the "memoryless" property, where actions taken in a state depend only on that state and not prior history. We then recursively solve the problem as a Dynamic Programming model utilizing Bellman's Equation with no discount to determine the optimal action (Bellman 1956). It is worth noting that this is neither direct discrete optimization of a design's characteristics, nor Pareto combinatorial optimization that yields a non-dominated set of point solutions. The three methods will be compared using example problem data.

THE DECISION PROBLEM

We assume a system consists of several subsystems. Each subsystem has alternative technologies and design options to meet the system requirements. A final point design consists of exactly one choice of an option for each subsystem. In the general case, not all the subsystems will be present in the final design, so the option of "none" is a realization option for each subsystem.

During development, technologies and design options are selected or rejected, subsystems are designed, and integrated in a task network organization leading to complete prototype production and testing. Subsystem design and integration takes time and incurs cost, and the time and costs depend on the technology and design option choices. The end system performance, production cost, and the development time and cost all depend on: (1) which subsystem technologies and design options are chosen, and (2) when the selection/rejection decisions are made.

During development, information regarding the distribution of the external factors changes. Over time, as more data becomes available, estimates of the means change and uncertainty generally

decreases – although it is possible that uncertainty can increase.

The decision problem is to select or reject subsystem options at the appropriate milestone or epoch to achieve "best value", the Contribution-to-Design at the end of the development program, despite adapting to changes in the externalities.

For the illustrative example below, we will assume that we only have three total system designs {D1, D2 and D3} from any unique and allowable combination of subsystem option. These three designs enumerate into seven "Set Solutions" (SS1 to SS7). Additionally, we also show the possibility of skipping design work.

In Figure 1, the PD milestones/Epochs are arranged in time order, such that Epoch A is first and Epoch Ω is last. The set solutions are all of the possible combinations of the three designs and are shown up to Epoch Ω. At Epoch Ω, the last epoch, the final design must be selected, so only the designs are shown. This is a visual representation of the Set-Based Design Process for considering which designs to develop. The colored arcs below represent one (yellow) or two (red) designs from not developing a design in the previous epoch. In most programs, these are possible but unlikely. For example, SS1 in B going to SS2 in Γ requires catching up all the work for D2 missed before B.

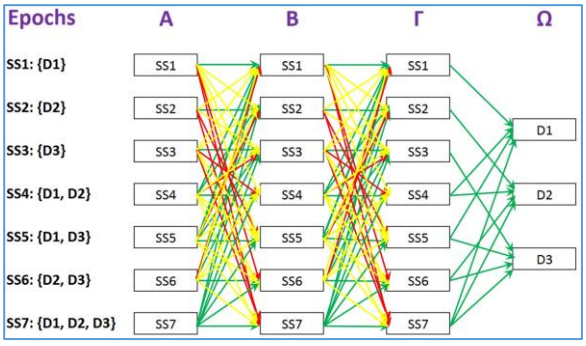

Figure 1: Example Problem – Total Set Solutions

DECISION SITUATIONS AND DESIGN DECISION STRATEGIES

There are four macro decision situations the framework supports (a – d below):

When new information regarding the distributions of the externalities becomes available,

When an option enters the critical path, i.e., if the design and integration of the option does not begin immediately, but has begun later, it increases the total development time,

When an option reaches the point at which keeping it under consideration, development or integration begins to incur costs at a higher rate, and

When new information indicates that a new option is no longer feasible.

Situation (a) involves reconsideration of the entire set-solution. Situations (b) and (c) involve only the decision to keep (b) or reject (c) the specific option. Situation (d) requires the problem to be restructured without the option by removing any system design choices that included the option.

Design decision strategies for a point solution are restricted to either modifying the current point solution or replacing it with a new point solution. For set solutions, the choices are more resilient. One can choose to neck down (i.e., reduce the set size), open up, or completely modify the set solution.

The following are examples of design decision strategy changes:

Neck-Down Example: Epoch A: SS7 to Epoch B: SS6. D2 dropped at Epoch B.

Open-Up Example: Epoch A: SS3 to Epoch B: SS5. D1 added at Epoch B with recovery costs.

Complete Modify Example: Epoch A: SS4 to Epoch B: SS3. D1 and D2 dropped and D3 added at Epoch B with recovery costs.

FRAMEWORK DIMENSIONS

Epochs aka Milestones (E) – Discrete Time points from A to Ω where:

Ω = total number of epochs (Ω =3 in the illustrative example):

t is the epoch index

Design Data is updated, reviewed and then used to make forward programmatic decisions from A, $B, \Gamma \ldots t+1, \ldots \Omega$

System Designs (Δ) – the allowable individual system designs possible from the system trade space from 1 to D where:

D = total number of allowable system designs (D=3 in the illustrative example):

d is the design index

A design must include exactly one option for every subsystem or technology under consideration in the trade space

Each System Design, Δ_d is a unique combination of technology options of exactly one per subsystem and is both a point solution and a singleton set solution

Other set solutions are the combinations of multiple system designs (see Figure 1)

Subsystems or Technology types (SS) – The discrete technology subsets that are required to frame a complete system choice where:

I = total number of technology subsets or subsystems (I =3 in the illustrative example)

i is the technology subset or subsystem index

Options (O) – for every technology subset or subsystem there are a set of options available where:

J_i = total number of options per i^{th} subsystem (J_1, J_2, J_3) = (2,2,2) in the illustrative example)

j is the i^{th} subsystem option sub-index

K = total number of options in the trade space (K =6 in the illustrative example)

k is the iterative vector transformation index taken from of the (i,j) option pairings

TOTAL DEVELOPMENT COSTS

The framework calculates Development Costs for every option epoch to epoch. Recovery Costs are calculated for options not developed in prior phases. Finally, System Integration costs are calculated for all phases. The framework considers the following:

Options may share certain costs with other options,

Reduce or Increase System Integration costs dependent on the shared development,

Recovery Costs are calculated for all options where the timeline permits option recovery,

There are no recovery costs if the new Set Solution is the same as the old,

Recovery costs are lower if the new Set Solution shared development with the old, and

Recovery costs are higher if development needs to be made up.

Contribution-to-Design (CTD)

We assume that a PD program will consider an initial set solution at the beginning and then will potentially modify that set solution at the different program milestones/epochs based on new information as time progresses and uncertainty clears. The Contribution-to-Design (CTD) is calculated for each set solution at each epoch. The framework supports the natural PD cycle where subsystem options are typically developed independently early and then require system integration later. The following Table 1 shows the relationships between the framework dimensions as PD advances through the epochs and phases.

Epochs	Key Dimensions	Value Calculations	Development Costs (DC)
A	Options	Set Solution CTD	Options Dev (A to B)
B to (Ω - 1)	Options	Set Solution CTD	Options Dev + Recovery Costs (B to B+1)
Ω - 1	Options	Set Solution CTD	Options Dev + SI Costs (Ω-1 to Ω) *** Final Design Set
Ω	Designs	Point Design Value	Options SI Costs (Ω to TP) for the Optimal Design (Δ*)
Test/Production (TP)	Final Design	NA	NA
Phases			**Cost Control Equations for Phase (E^i to E^{i+1})**
E^A to E^B			$\sum_{k=1}^{K} DC_k$ <= Phase Development Budget
E^B ... to ... $E^{\Omega-1}$			$\sum_{k=1}^{K} DC_k$ <= Phase Development Budget
$E^{\Omega-1}$ to E^{Ω}			$\sum_{k=1}^{K} DC_k + \sum_{d=1}^{D} \Delta_d SI$<= Phase Development Budget
E^{Ω} to E^{TP}			Final Design SI Costs <= Pre-Test Development Budget

Table 1: Framework Dimensional Relationships

In Table 1, we show what the key dimensions, values and costs are that the framework must track and calculate to data populate the Markov Decision Process algorithm. We also show, what the required cost balance equations are for each phase of work, i.e. epoch to epoch. At Epoch **Ω**, we down select to the final design. At **Ω**, the option development is complete, so the point values of the designs are all that remains. It is assumed that only SI costs remain at this point in the process.

General Control Equations: (1) Reject options development that exceeds the phase budgets, and (2) Select exactly one Design at Epoch Ω.

THE CONTRIBUTION-TO-DESIGN FUNCTION: OPTION, SUBSYSTEM AND SET

The Contribution-to-Design (CTD) for an option includes both the performance/burden to target difference of the technology/option and the expected variance of the developing technology/option. The framework formulation measures the expected confidence of the specific technology/option to meet Design Readiness Levels during the phases. The Design Readiness Level is an expansion of the DOD's Technical and Material Readiness Levels. In the context of framework, the *Design Readiness Level* reflects the maturity of the design relative to each of required target levels in a program. Without loss of generality, in the illustrative example, we employ three levels for design readiness: 1-Least Ready; 2-Somewhat Ready; and 3-Fully Ready. While the estimated confidence is based on estimates, the framework allows for continuous updating as uncertainty in the estimates lessens over time. With the inclusion of variance, uncertainty can be properly modeled. Additionally, this approach has significant computational advantages since it permits meta-heuristic optimization techniques. We calculate CTD (s) for: options, subsystem and set solution.

This is the Contribution-to-Design formulation for an option at a specific Epoch looking forward:

$$\text{CTD} = \sum_1^n \omega_i \cdot P[Z_i] \qquad (1)$$

where P is the probability lookup of the Z_i value (defined below), ω_i is the weight of the externality (performance or burden parameter), and i is the index for all n externalities.

$$\sum_1^n \omega_i = 1 \qquad (2)$$

$$Z_i = \{(\mu_i - L_i)/\sigma_i\} \text{ where:} \qquad (3)$$

L = Performance or Burden Requirement Target Value

$\{\mu, \alpha\}$ = System or Sub-system externality probability distribution

In Table 2, we show an example of Option 1's individual Contribution-to-Design calculations for meeting Readiness Levels 1 and 2. In this example we "weight" the three metrics (performance, physical weight burden and AUPC burden). The Contribution-to-Design is then a weighted value of the three probability measures.

Epoch A Option 1	Performance	Weight Burden	AUPC Burden	CTD
Weight	0.3	0.3	0.4	
Current Mean	350	350	47000	
Current SD	50	100	10000	
DRL 1 Target	250	400	45000	
DRL 2 Target	300	390	44000	
P (X > DRL 1 Tgt)	0.9772	0.6915	0.4208	0.6689
P (X > DRL 2 Tgt)	0.8414	0.6554	0.3821	0.6019

Table 2: Individual CTD Calculation

For Set Solutions with more than one option per subsystem, it is expected that the multiple options reduce total design uncertainty should a single option fail to meet targets.

This is the cornerstone of SBD and has been effectively utilized for over twenty years, even if not quantitatively proven. For the framework, we assume that the multiple option designs are independent since we only calculate the probabilities of exceeding targets of the readiness levels. Given independence, the Contribution-to-Design formulation for multiple options in a subsystem is:

$$CTD_{i,RL_t \to RL_{t+1}} = 1 - \prod_{j=1}^{m_i} \left\{ 1 - \sum_{RL_t=1}^{3} \{1\}_{x_{i,j,t,RL_t}} Pr_{i,j}\{RL_t \to RL_{t+1}\} CTD_{i,j,RL_t \to RL_{t+1}} \right\}$$

$\{1\}_{x_{i,j,\Omega,RL_t}}$: 1 if option j of subsystem i is selected and its current state in Ω is RL_t, 0 otherwise.

(4)

where CTD_i is the subsystem CTD for the i^{th} subsystem. j is the index for the option members, from 1 to m, where m is the total number of options in the same subsystem that are in the set solution.

For the example problem, we will use the data from Table 3 for the third subsystem in Epoch B moving forward to Design Readiness Level of 3. We assume the set solution has options 5 and 6:

CTD (Subsystem 3, Epoch B to Epoch Γ, DRL 3) = $\{1 - ((1 - 0.7) \cdot (1 - 0.1))\} = \{0.73\}$.

The total CTD for the set solution must then consider the individual subsystem CTD's at the epochs. Depending on program situation, it may be important to weight the subsystem CTD's differently. For example, an automotive program may value its engine subsystem higher than its entertainment system for its sports car while it reverses that weight value for its minivan. The CTD for the entire set solution at an Epoch moving to given Design Readiness Level during the next phase is:

$$CTD = \sum_1^I \{Wt_i \cdot CTD_i\} \qquad (5)$$

where I = the number of subsystems, i is the subsystem index, and Wt is the relative weight of the subsystems value.

For the example problem, we will use data from Table 3 at Epoch B for Design Readiness Level 3 and an arbitrary weight vector of {0.4, 0.4, 0.2} for the subsystems. The example set here includes all options except for Option 1.

CTD (Set Solution, Epoch B, DRL 3) = $((0.4 \cdot 0.9) + (0.4 \cdot 0.73) + (0.2 \cdot 0.73)) = 0.81$.

Epoch A	1	2	3	4	5	6	Epoch B	1	2	3	4	5	6
Action	1	1	1	1	1	1	Action	0	1	1	1	1	1
State DRL	1	1	1	2	1	2	State DRL	1	2	2	2	2	2
P(DRL => 1)	1	1	1	1	1	1	P(DRL => 1)	1	1	1	1	1	1
P(DRL => 2)	0.6	0.7	0.5	1	0.7	1	P(DRL => 2)	1	1	1	1	1	1
P(DRL => 3)	0.3	0.2	0.2	0	0.3	0	P(DRL => 3)	0.2	0.9	0.7	0.1	0.7	0.1

Table 3: Example Action, State and Transition Matrix

In Table 3, we show examples of the required action and state spaces to enable the algorithm. The current Design Readiness Levels are in Row 3 and Rows 4 to 6 show the probability of the options moving up if invested in for the phase development to the next epoch. Any probability distribution function is allowable as we are only concerned with the likelihood state changes.

THE VALUE FUNCTION: STATE TO STATE CTD IS THE MDP REWARD

This is the core of the research development. We begin at the end, Epoch Ω, knowing that we must down select to a single integrated prototype design. We accept that there may be some modification, rework and additional SI during testing. The value function at the end is simply a single weighted multi-attribute value of each design. The key insight of the research is that we only need to solve the actions of the options themselves and not the designs. The designs under development prior to full system integration are an extension of the option developments prior to the final epochal decision to design down select.

The Dynamic Programming (DP) back solve is initialized by calculating the CTD values to seed the MDP and its reward function R beginning with the last epoch just before the system integration. We use the CTD values, which we calculate at each state space, to be a "black box" value from which we can determine a value change, the reward R, from state to state. This creates a set of stochastic automata with utilities for DP solving.

MDP(s) include: (1) a set of possible world spaces $\{S\}$, (2) a set of possible actions $\{A\}$, (3) a real valued reward function $R(S, A)$ and a description T of each action's effects in each state. For the framework, this is a weighted selection of all the options that considers not only the performance and burdens of each of the options, but also their likelihood to meet their performance requirements and burden budgets.

During the phases, we consider budgetary controls to not exceed the phase budget, which is standard DOD policy. We consider every possible action that does not violate the budget. The recursive DP solves for the optimal action. We utilize Bellman's stochastic balance equations to solve for the optimal initial action [4]. The actual value optimized recursively is the Expected Value of all CTD improvements. This modified optimization approach allows us to focus on which options are invested in for the optimal actions determined at each Epoch. We further assume the memoryless Markov Property: the effects of an action taken in a state depends only on that state and not on the prior history and we apply no discount factor.

THE RESEARCH EXAMPLE PROBLEM

Figure 2 shows the goals hierarchy and value weights for the example problem. There are two subsystems and three metrics for the problem itself. For this simple problem we are able to use the straight forward Multi-Attribute Utility model to determine both the discrete single point optimal design and to also show the full combinatorial optima solution. Figure 3 shows that D1 is the single best design as it has the best total weighted utility.

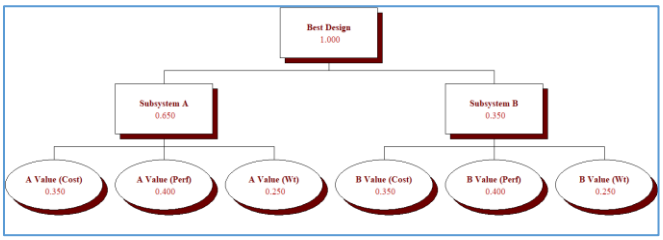

Figure 2: Problem Goals Hierarchy

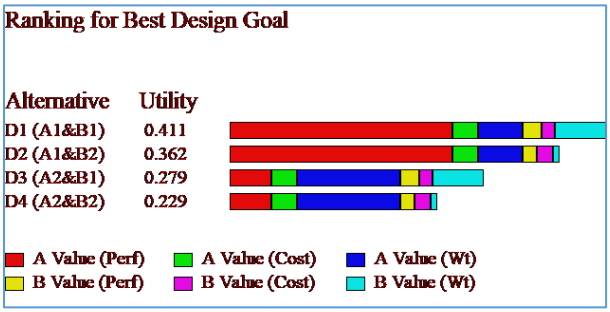

Figure 3: Ranking for Best Design

Table 4 shows the CTD Based Expected Values that come from the main problem. These Expected Values assume that the Designs would follow the typical PD process for point based design, i.e. the options of the design are developed continuously. Additionally, we show the CTD values at the metric level so we can determine if there is any pareto dominance between the designs.

Designs	A Values			B Values		
	Perf	Weight	Cost	Perf	Weight	Cost
D1 (A1&B1)	0.93175	0.30000	0.12500	0.14950	0.63600	0.12500
D2 (A1&B2)	0.93175	0.30000	0.12500	0.11700	0.09200	0.14600
D3 (A2&B1)	0.17750	0.69200	0.12500	0.14950	0.63600	0.12500
D4 (A2&B2)	0.17750	0.69200	0.12500	0.11700	0.09200	0.14600

Table 4 – CTD Based Expected Values (Deconstructed down to metric)

Green represents the maximum metric value. Although D1 almost dominates D2, and D3 almost dominates D4, the B2 option is strongest in cost value, which keeps D2 and D4 as combinatorial

optima. Thus, the combinatorial solution shows no complete pareto dominance for any design. The program team would be left to decide its best design solution to take forward.

The process for defining the optimum or optima in single point approaches focuses on selecting a design initially and then developing it. The framework approach rather determines an optimal action to initiate a set solution from an initial optimal action. That action is based stochastically, to develop a set of options that reduces developmental risk and allows for uncertainty in the process. We compare the results from these methods to the results from proposed framework method in the next section.

FRAMEWORK STATE AND ACTION SPACES

The State Space is the set of all possible states for the problem, which is every possible combination. The individual state's dimensions are carried in a vector of: K (total number of options) × (number of parameters/metrics) + epoch. The non-epoch cells hold the Design Readiness Levels that can vary, but for the example we use exactly three levels per option/metric combination. The following Table 5, shows the origin and end states. The state space contains four possible designs. The possible maximum size of the example state space is the product of all option Design Readiness Levels (3^{12}) and the number of Epochs (4) or 2,125,764.

Options	A1			A2			B1			B2		
Metrics	P	W	C	P	W	C	P	W	C	P	W	C
$State_A$	1	1	1	1	1	1	1	1	1	1	1	1
Options	A1			A2			B1			B2		
Metrics	P	W	C	P	W	C	P	W	C	P	W	C
$State_\Omega$	3	3	3	3	3	3	3	3	3	3	3	3

Table 5 – Notional Origin and End States

The MDP Network for our example, is much sparser since we do not start or end at the extremes and we do not have state to state arcs for every possibility. The actual example problem state space is less than 1,000.

In Table 6 below, the Action Space is a matrix of all possible options length that represents whether an option is invested in (1) or not invested in (0) for the next phase's development. Additionally, the Action Space can be expanded to cover skipped investments. A skipped investment would be marked by a (2) and the Action Space would then be a maximum in size of 81 possible actions. This is a rather rare occurrence in the real world, so it would be easy enough to just add the small set of skip actions for computation purposes.

Actions	A1	A2	B1	B2
a_1	0	0	0	0
a_2	0	0	0	1
a_3	0	0	1	0
a_4	0	0	1	1
a_5	0	1	0	0
a_6	0	1	0	1
a_7	0	1	1	0
a_8	0	1	1	1
a_9	1	0	0	0
a_{10}	1	0	0	1
a_{11}	1	0	1	0
a_{12}	1	0	1	1
a_{13}	1	1	0	0
a_{14}	1	1	0	1
a_{15}	1	1	1	0
a_{16}	1	1	1	1

Table 6 – Example Problem Action Space

The Transition arcs are stochastic. Most of the problem model's transitions are two choices of either move up one or remain the same Design Readiness Level for the individual options, although we did include some other transitions to test the model's robustness. The actual set solution transitions typically numbered 16 or more are shown below in Table 7. The Rewards as stated earlier, are state to state changes in the CTD.

State	Opt Index				Epoch	Pred
Index	A1	A2	B1	B2	In	Node
1	13	5	13	3	A	Origin
1	13	5	13	3	B	1
2	13	5	13	5	B	1
3	13	5	13	11	B	1
6	13	5	14	3	B	1
7	13	5	14	5	B	1
8	13	5	14	11	B	1
21	13	14	13	3	B	1
22	13	14	13	5	B	1
23	13	14	13	11	B	1
26	13	14	14	3	B	1
27	13	14	14	5	B	1
28	13	14	14	11	B	1
81	14	5	13	3	B	1
82	14	5	13	5	B	1
83	14	5	13	11	B	1
86	14	5	14	3	B	1
87	14	5	14	5	B	1
88	14	5	14	11	B	1
101	14	14	13	3	B	1
102	14	14	13	5	B	1
103	14	14	13	11	B	1
106	14	14	14	3	B	1
107	14	14	14	5	B	1
108	14	14	14	11	B	1

Table 7 – Transition Matrix Example

MDP WITH DP SOLVE

Once the MDP is seeded with the CTD values, we can conduct the DP recursion. First, we calculate from Epoch *Γ* to Epoch Ω the Expected Values of the CTD's for every possible action because each state to state arc for each action has an associated probability. Once that is accomplished, we have the optimal action and its Expected Value calculated for every Epoch *Γ* state to its children Epoch Ω states.

For the example problem, this corresponds to measuring approximately 2,000 arcs to determine the Optimal Actions and Expected Values for the 108 Epoch *Γ* states. We continued recursively to solve the Epoch *B* to Epoch *Γ* (approximately 500) arcs to find the optimal actions for the 24 Epoch *B* states and we pass the previous best Expected Value for those Actions. We then repeat the same process to calculate the Optimal Action from the MDP Networks origin node at Epoch *A*. Table 8 below is a small subset of the more than 2,500 Expected Value/Optimal Action sets of calculations.

	A to B	B to Γ	Γ to Ω	Budget - $M
Model 1	a16	a16	a13	72
Model 2	a12	a12	a11	53
Model 3	a8	a8	a7	46
Model 4	a12	a12	a11	53
Model 5	a12	a12	a11	53
Model 6	a12	a12	a11	53
Model 7	a4	a53	a11	51

Table 8 – Best Action and Expected Value Example

SENSITIVITY ANALYSIS OF THE FRAMEWORK MODEL

Multiple Scenario Models were created to conduct sensitivity analysis on the proposed Framework Model. Figure 4 below shows the variant models.

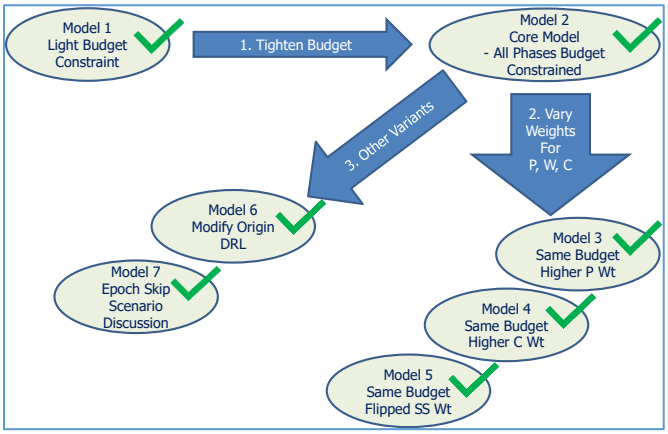

Figure 4 – Variants to Test Sensitivity

Table 9 below the optimal actions.

Arc Number	Epoch In	Node In	Node Out	Epoch Out	Γ_Ω Best Action	Γ_Ω CTD EV	B_Γ Best Action	B_Γ CTD EV	A_B Best Action	A_B CTD EV
Origin				A					a12	0.9927
1	A	1	1	B			a12	0.9850		
6	A	1	8	B			a13	0.9898		
7	A	1	21	B			a12	0.9866		
17	A	1	87	B			a8	0.9915		
24	A	1	108	B			a12	0.9927		
25	B	1	1	Γ	a7	0.9365				
28	B	1	6	Γ	a6	0.9442				
31	B	1	21	Γ	a11	0.9614				
36	B	1	28	Γ	a10	0.9786				
37	B	1	81	Γ	a7	0.9614				
45	B	1	103	Γ	a4	0.9786				

Table 9 – Sensitivity Analysis - Framework Models

Model 1 had a lightly constrained budget and correctly picked all options where it could. In Model 2, the budget was tightened to specifically force selections between options and it did so correctly. In model 3, we adjusted the performance weight much higher, and the model went for the development of option A2 as it became more attractive than A1. In Model 4, we adjusted the cost weight higher and it went back to the same actions in Model 2, although the EV numbers were different. If the cost weight would have been forced even higher a change in actions would have occurred. In Model 5, we flipped the subsystem weights. Although the numbers did change, it was not enough to alter the actions. Model 6 modified the network origin and had over 90% different arcs. The same modeling structure was applied as in Model 2, so that actions mimicked Model 2. However, the numbers for the EV were completely different as the state locations were vastly different. Model 7 covered a skip and recovery. In this version, we modeled a later start, but with more attractive metrics and better budget. The model did take the skip and recovery for this unusual scenario. Generally, development is required to reduce uncertainty, but the framework can handle budget skips.

SUPPORTING FUNCTIONS NEEDED TO IMPLEMENT THE MODEL

Each externality or parameter/metric must include a target value and a random value (RV) distribution of the technology/option to determine the confidence of the individual design sets (singletons), and then ultimately the multiple design sets. That includes: performance and burden metrics.

Each design must calculate the development costs from epoch to epoch and the recovery costs from the previous epoch to catch up if the set solution did not include that option previously. Additionally, reductions and increases associated with shared development and SI costs must be calculated.

Finally, although not shown here in the example problem, the weights of the externalities themselves are also RV's. This framework can be extended with a Monte Carlo simulation. The simulation would create a data set of random externality weights from the weight RV's. Each one of these could then be solved individually to see the impact on the Set Solutions from epoch to epoch.

COMPARATIVE ANALYSIS OF THE THREE ANALYTICAL METHODS

We cannot completely compare the three methods since the framework method employs a DP recursion to calculate optimal actions vs. the single point and combinatorial forward models. However, we can consider what the forward expected CTD would be for all actions when executing the methods going forward. Developing the individual designs from A to Ω is a pure set of a_{11} actions for Design 1, a_{10} for Design 2, a_7 for Design 3, and a_6 for Design 4. The optimal set of actions, recursively solved in the framework solution are: a_{12}, a_{12}, and a_{11}. See Table 10 below for the comparisons.

	A to B	B to Γ	Γ to Ω	System EV	Budget - $M
Design 1	a_{11}	a_{11}	a_{11}	0.953602	47
Design 2	a_{10}	a_{10}	a_{10}	0.946763	43
Design 3	a_{7}	a_{7}	a_{7}	0.947511	40
Design 4	a_{6}	a_{6}	a_{6}	0.940672	36
SBD Framework	a_{12}	a_{12}	a_{11}	0.963733	58

Table 10 – Forward CTD Expected Values for Comparing Analytical Methods

From a pure quantitative sense, the SBD Framework finds the best expected value. Of note, Action a_{12} is the same as jointly executing a_{10} and a_{11}, which are the two best single point design approaches until dropping down to a_{11} in the final phase. This approach reduces the risk of carrying two designs through the first two phases until uncertainty is reduced.

Perhaps the biggest issue regarding the use of SBD is the potentially higher cost associated with it. For this example, the SBD Framework approach may cost an extra $11M to execute. That is the immediate trade-off you take with SBD. However, further factors support the SBD Framework yielding superior design results. The first is that Production costs are at least one order of magnitude higher than R&D costs, let alone Lifecycle costs which are typically a magnitude higher than Production costs. Given that, budget increases for R&D to support SBD make good sense.

Secondly, the U.S. DOD recognizes this developmental risk and often awards two identical design contracts in the hopes that two contractors are better than one. They presume independence in the development. Thus, the typical action would be to pay $94M for two contractors which is $36M more in the example problem. However, contractors working on different options is inherently more independent than two contractors working on the same options. While their thought processes may differ, both contractors are subject to the same physics and the same engineering issues when developing the same design. Further study on how independent designs are in like contracts, is warranted.

NEXT STEPS

Our framework process, which calculates the probability density functions and values for the Contribution-to-Design function, needs efficiency improvements to handle large scale models. Use of Design Structure Matrices (DSM) in parallel with this framework can further enhance the scalability of the framework.

CONCLUSIONS

The literature review documented the current state of Set-Based Design as a resilient Product Development process that has proven qualitative results over the past two decades for many Product Development programs. Quantitative processes with design optimization traditionally focus on point solutions. Sophisticated, combinatorial optimization has increased insight into design uncertainty, but still provide more sophisticated point solutions. There is a significant need to have a Set-Based Design quantitative solution structure that balances both optimality in a point solution sense with variability to capture set solutions in clustered set solutions [5]. Resilience in the Product Development process is supported by utilizing Set-Based Design to create multi-design sets that increase confidence to attain meeting design target requirements.

The initial mathematical framework proposed in this paper shows a method to combine performance and burden information from competing design elements into a Contribution-to-Design Function that can be optimized [6]. Specifically, Contribution-to-Design is an optimality structure formulating Set-Based Design value which is not point design optimization in the current vernacular. The value is directly associated with maintaining design sets to directly increase design resilience in the face of design uncertainty. This directly addresses objective 1. A cost structure that details developmental costs, design recovery costs and

system integration costs from epoch to epoch is proposed. By updating the cost structures and performance estimates at each epoch a design fluidity supports set adjustments at epochal decision points. This also supports the decisions as to when to modify the sets, which addresses objectives 2 and 3. Key linkage between the Set-Based Design process and the usage of Process Design Structures Matrices to support design process resiliency was postulated for Navy ship design [7]. Further research expanding the framework to integrate Design Structures Matrices would allow scalability. The bottom line is: Set-Based Design improves mathematical optimization to make better trade space decisions. This is done by expressing set solution value and incorporating uncertainty into the mathematical optimization.

REFERENCES

[1] B. Martin, "Arleigh Burke Destroyers: Additional Analysis and Oversight Required to Support the Navy's Future Surface Combatant Plans", G. Audit. Washington, DC, U. S. Government Accounting Office, 2012.

[2] D. Singer, "What Is Set-Based Design?" Naval Engineers Journal (4): 13, 2009.

[3] S. Rapp, "Product Development Resilience through Set-Based Design", Dissertation, Digital Commons @ Wayne State University, 2017.

[4] R. Bellman, "Dynamic programming and Lagrange multipliers." Proceedings of the National Academy of Sciences, 42(10), 767-769, 1956.

[5] G. Avigad, "Set-Based Concept Selection in Multi-Objective Problems: Optimality vs. Variability Approach." Journal of Engineering Design 20:25, 2007.

[6] S. Rapp, "Product Development Resilience through Set-Based Design", Systems Engineering Journal, 2018.

[7] N. Doerry, "Using the Design Structure Matrix to Plan Complex Design Projects", ASNE Intelligent Ships Symposium ASNE Intelligent Ships Symposium. Philadelphia, PA, ASNE: 15, 2009.