VT-TE-19-196

ACQUISITION RESEARCH PROGRAM
SPONSORED REPORT SERIES

Dynamic Contracting of Verification Activities by Applying Set-based Design to the Definition of Verification Strategies

29 August 2019

Dr. Alejandro Salado

Peng Xu

Virginia Tech

Disclaimer: This material is based upon work supported by the Naval Postgraduate School Acquisition Research Program under Grant No. HQ0034-18-1-0002. The views expressed in written materials or publications, and/or made by speakers, moderators, and presenters, do not necessarily reflect the official policies of the Naval Postgraduate School nor does mention of trade names, commercial practices, or organizations imply endorsement by the U.S. Government.

Prepared for the Naval Postgraduate School, Monterey, CA 93943.

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL

The research presented in this report was supported by the Acquisition Research Program of the Graduate School of Business & Public Policy at the Naval Postgraduate School.

To request defense acquisition research, to become a research sponsor, or to print additional copies of reports, please contact any of the staff listed on the Acquisition Research Program website (www.acquisitionresearch.net).

Acquisition Research Program
Graduate School of Business & Public Policy
Naval Postgraduate School

Executive Summary

This report describes recent research in support of acquisition programs that use verification activities to elicit evidence of contractual fulfillment. Thus, the importance of adequately defining verification activities in any acquisition program is unquestionable. Its significance extends beyond contracting though. The biggest portion of the development financial budget is spent in executing verification activities and verification activities are the main vehicle in discovering knowledge about the system, which is key to reduce development risk. In current practice, a verification strategy is defined at the beginning of an acquisition program and is agreed upon by customer and contractor at contract signature. Hence, the resources necessary to execute verification activities at various stages of the system development are allocated and committed at the beginning, when a small amount of knowledge about the system is available. However, contractually committing to a fixed verification strategy at the beginning of an acquisition program fundamentally leads to suboptimal acquisition performance. Essentially, the uncertain nature of system development will make verification activities that were not previously planned necessary and will make some of the planned ones unnecessary. Therefore, dynamic contracting of verification activities is necessary to guarantee optimality of acquisition programs in this area.

In order to cope with these challenges, this research project addressed the main question of whether set-based design can enable the execution of dynamic contracts for verification strategies, ultimately resulting in more valuable verification strategies than current practice. In particular, this research project had the following objectives: (1) Given an optimal verification strategy at a point in time, generate a set of optimal future verification paths; and (2) Conduct a comparative analysis between set-based design for verification and a benchmark. This research employed a combination of a computational framework and a simulation tool. The hypotheses were tested on a notional Earth observation satellite instrument, representative of those of interest to the Air Force.

By fulfilling the research objectives, this research is anticipated to promote higher early safety and efficacy of commercial products and public services. While an application for the Air Force has been used as a test case, the methodologies and insights provided in this work can be applicable to a broad range of systems that are subjected to limited verification: other defense systems, space systems, aeronautics, automotive systems, manufacturing systems, electronic products, civil infrastructure, public health systems, or transportation systems.

The research has already resulted in one published paper for the 2019 Acquisition Research Symposium, one published paper for the 2019 INCOSE International Symposium, and one published paper in Wiley's Systems Engineering journal. Two other journal papers resulting from this work are currently under preparation and will be submitted before the end of 2019.

VT-TE-19-196

ACQUISITION RESEARCH PROGRAM SPONSORED REPORT SERIES

Dynamic Contracting of Verification Activities by Applying Set-based Design to the Definition of Verification Strategies

29 August 2019

Dr. Alejandro Salado

Peng Xu

Virginia Tech

Disclaimer: This material is based upon work supported by the Naval Postgraduate School Acquisition Research Program under Grant No. HQ0034-18-1-0002. The views expressed in written materials or publications, and/or made by speakers, moderators, and presenters, do not necessarily reflect the official policies of the Naval Postgraduate School nor does mention of trade names, commercial practices, or organizations imply endorsement by the U.S. Government.

THIS PAGE LEFT INTENTIONALLY BLANK

Table of Contents

THIS PAGE LEFT INTENTIONALLY BLANK

List of Figures

THIS PAGE LEFT INTENTIONALLY BLANK

List of Tables

THIS PAGE LEFT INTENTIONALLY BLANK

Background

Verification activities, which usually take the form of a combination of analyses, inspections, and tests, consume a significant part, if not the biggest part, of the development costs of large-scale engineered systems (Engel, 2010). Verification occurs at various integration levels and at different times during its life cycle (Engel, 2010). Under a common master plan, low level verification activities are executed as risk mitigation activities, such as early identification of problems, or because some of them are not possible at higher levels of integration (Engel, 2010). Therefore, a verification strategy is defined "aiming at maximizing confidence on verification coverage, which facilitates convincing a customer that contractual obligations have been met; minimizing risk of undetected problems, which is important for a manufacturer's reputation and to ensure customer satisfaction once the system is operational; and minimizing invested effort, which is related to manufacturer's profit" (Salado, 2015). Essentially, verification activities are the vehicle by which contractors can collect evidence of contractual fulfillment in acquisition programs.

In current practice, a verification strategy is defined at the beginning of an acquisition program and is agreed upon by customer and contractor at contract signature. Hence, the resources necessary to execute verification activities at various stages of the system development are allocated and committed at the beginning, when a small amount of knowledge about the system is available (Engel, 2010). However, the necessity and value of a verification activity cannot be measured independently of the overall verification strategy (Salado & Kannan, 2018b). Instead, the necessity to perform a given verification activity depends on the results of all verification activities that have been previously performed (Salado & Kannan, 2018b). For example, testing the mass of a component is considered more necessary if a previous analysis has shown low margin with respect to the success criterion than if the analysis has shown ample margin. Thus, contractually committing to a fixed verification strategy at the beginning of an acquisition program fundamentally leads to suboptimal acquisition performance. Essentially, the

uncertain nature of system development will make verification activities that were not previously planned necessary and will make some of the planned ones unnecessary (Salado & Kannan, 2018b). The former can be handled through change requests (CR) but they require unplanned financial investments. The latter can be recovered in a few cases through negative change requests but, in general, they imply a waste of the financial investment because the investment has been committed to the contractor.

In this context, dynamic contracting of verification activities becomes necessary to guarantee optimality of acquisition programs in this area (Xu & Salado, 2019). Instead of contracting a predefined set of activities at the beginning of a project, the necessity and contracting of each verification activity (or subsets of them) are evaluated and executed as the system development progresses (Xu & Salado, 2019). Set-based design has been proposed as part of this research to support such contracting approach (Xu & Salado, 2019). Informed by the benefits of set-based design in conceptual design (Singer, Doerry, & Buckley, 2009), an overall set of verification activities is considered, but not contracted, at the beginning of a project. A vector of investment opportunities indicates the development stages in which verification activities may be contracted and executed. Based on their results, the set of remaining verification paths to the end of the system development is updated (Xu & Salado, 2019).

Literature Investigation

Set-based Design

Verification strategies are defined in current practice at the beginning of an acquisition program and are agreed upon by customer and contractor at contract signature, when a small amount of knowledge about the system is available (Engel, 2010). Such lack of knowledge in early design activities motivated the emergence of set-based design (Bernstein, 1998). Set-based design is built on the principle of working simultaneously with a plethora of design alternatives, instead of converging quickly to a single option (Bernstein, 1998). As the knowledge about the system increases, suboptimal alternatives are discarded until a preferred one remains (Bernstein, 1998). A key aspect is that discarding is not an activity at a given point of time, like a traditional trade-off, but a time-continuous activity that occurs as new knowledge is available (Bernstein, 1998). A formal formulation of set-based design and how it make product development resilient against changes in external factors is given in (Rapp, Chinnam, Doerry, Murat, & Witus, 2018).

Set-based design has been successfully applied in the conceptual stages of naval systems (Singer et al., 2009), graphic industry products (Raudberget, 2010), automotive products (Raudberget, 2010), and aeronautic systems (Bernstein, 1998), among others. Historical analysis of the use of set-based design has shown that it inherently eliminates root causes of rework in system development (Kennedy, Sobek, & Kennedy, 2014). Researchers have integrated set-based design with tradespace exploration to further strengthen its value by leveraging the numerous solutions that tradespace exploration provides to generate the initial set (Small et al., 2018). However, empirical research about the implementation of set-based design in an industrial setting showed that there are some discrepancies as to how to operationalize the approach (Hansen & Muller, 2012). It remains to explore if this was an anecdotal episode or if it happens in general.

Verification as an engineering endeavor

Consider "a generic model of the expected utility $E\left[U_{S,p,t}\right]$ provided by a system S at time t with respect to a set of preferences P, as given in Eq. (1),

$$E\left[U_{S,P,t}\right] = F_U\left(S_A, B_t\left(S_A, t_n\right), P\right) \quad (1)$$

where S_A is a set of system characteristics, $B_t\left(S_A, t_n\right)$ is the belief at time t that those system characteristics will be exhibited by the system at a later time t_n, and F_U is a set of expected utility functions, associated with beliefs on those functions, that map system attributes, beliefs of system attributes, and preferences to expected utility" (Salado & Kannan, 2018b). In this context, a verification activity is one that "affects at least $B_t\left(S_A, t_n\right)$" (Salado & Kannan, 2018b). That is, a verification activity is one that, as a minimum, provides information about the system under development.

For the purpose of this research, two main characteristics of verification lead to the need for dynamic contracting of verification strategies. First, the value of each verification activity is not absolute, but depends on the results of prior verification activities (Salado & Kannan, 2018b). This means that the value of a verification activity cannot be determined individually, but in the context of the knowledge at the time of executing the activity. Therefore, the expected value provided by a verification activity evolves as a function of the results of previous verification activities. Second, although verification activities are objective, the confidence that they generate is subjective (Salado & Kannan, 2018b). This means that not only prior verification activities influence the value of a verification activity, but also the engineer or the team in charge of processing and interpreting the results of a given verification activity do so. Given the long development times necessary in some large-scale systems, it is common that the team in charge of executing verification activities towards the later stages of the system development is different from the team that planned those verification activities early in the lifecycle. Hence, changes in the perceived value of a verification activity is inherent to the nature of a large-

scale system development, under the assumption that the teams will change as the development progresses.

Mathematical models of verification strategies

In this report, a verification strategy is understood to be a set of verification activities organized as an acyclic directed graph (Salado & Kannan, 2018a). A verification activity is understood to be the collection of information about a specific aspect of the system under development (for simplicity we will call this a system parameter) and verification evidence refers to such information. Furthermore, it is assumed that the level of confidence in the correct performance of the system is shaped by the system architecture (e.g., maturity and coupling of the system's components) and the results of the various verification activities (Salado & Kannan, 2019).

Mathematically, this understanding is captured by "modeling the engineer's posterior belief distribution $\pi(\theta|\mathbf{s})$ based on his/her prior belief distribution $\pi(\theta)$ and the density function $f(\mathbf{v}|\theta)$, conditioned on the collected verification evidence $\mathbf{v}$", where θ is the system parameter that is verified and $\mathbf{v} \in V^*$ is a specific vector of verification results (or verification evidence) (Salado & Kannan, 2019). Using this mathematical framework, a verification strategy is modeled as a Bayesian network $BN = \Upsilon \cup A \cup B$, where (Salado & Kannan, 2019):

- $\Upsilon = (V, D)$ is a simple directed graph that captures the planned execution of verification activities. The set V is a set of verification activities and D is a set of tuples (a,b), with $a,b \in V$, that describes the relative order in which verification activities are planned to be executed (Salado & Kannan, 2018a).
- $A = (\theta_Z, D_\theta)$ is a simple directed graph that captures the properties of the system architecture, specifically the coupling between the different components forming the system, as well as their individual maturity. The set θ_Z captures the prior beliefs on the absence of errors in the system parameters and the information dependencies between those parameters are captured in the set $D_\theta = \{(a,b) : a,b \in \theta_Z, f(b|\mathbf{a}) \neq f(b)\}$.
- $B = (\{\theta_Z, V\}, D_\Upsilon)$ is a simple directed graph that captures the ability of the verification activities to provide information about one or more system

parameters, where $D_r = \{(a,b) : a \in \theta_z, b \in V, f(b \mid \mathbf{a}) \neq f(b)\}$.

Resulting graphs modeling verification strategies can be reduced to a combination of a finite set of patterns (Salado & Kannan, 2019). Identification of patterns may aid in interpreting the role of the various verification activities within a strategy. For example, a dynamic network (as will be used later in this report) indicates that certain activities may make some prior activities irrelevant once the new ones have been executed (Salado & Kannan, 2019).

It should be noted that the previous notation may not be followed throughout the report; it has been used here for consistency with the original source.

This modeling approach forms the basis for the mathematical model underlying the application of set-based design to the design of verification strategies presented in this report. The basic notation is represented in Figure 1. System parameters are denoted by θ_i and verification activities by V_i. Arrows represent information dependencies.

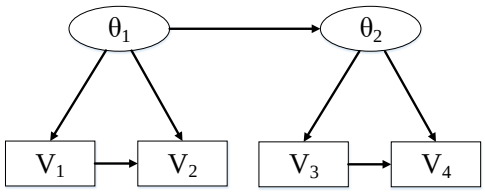

Figure 1. Example of modeling notation

In the example in Figure 1, θ_1 could represent, for example, the performance of a prototype, which is verified through an analysis V_1 and a test V_2 (such that the result of the analysis shapes the confidence on the expected result of the test). Such prototype performance shapes the confidence on the performance of the actual system θ_2, which is verified through verification activities V_3 and V_4.

Set-Based Design Applied to Verification Strategies

Concept

Note: *This section has been slightly adapted from a publication by the authors prepared, submitted, and published during the period of performance of this research (Xu & Salado, 2019).*

The approach developed in this research is graphically compared against the current paradigm for contracting verification activities in Figure 2. In the current paradigm (top part of the figure), a contract for a verification strategy is fixed at the beginning of the system development program. The strategy is defined by the black dots connected by the orange line, which represent the verification activities that will be executed throughout the system development.

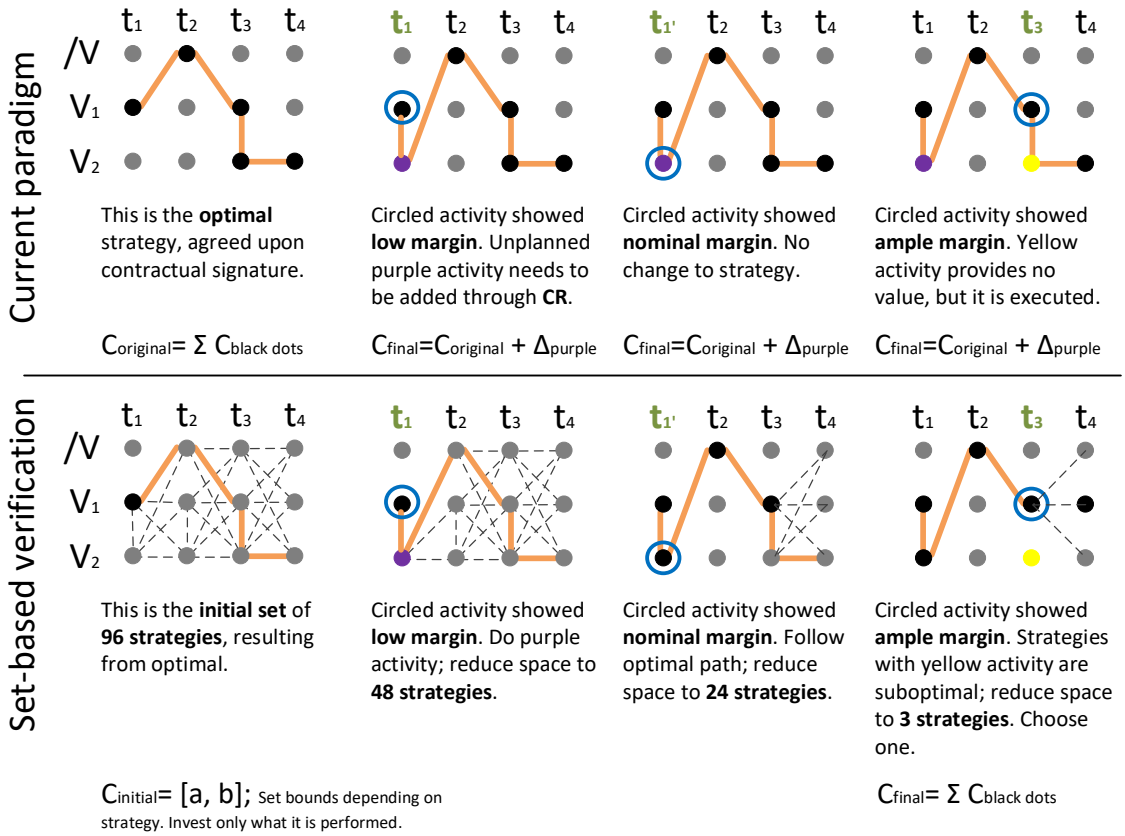

Figure 2. Current vs Set-based approaches for designing verification strategies

(*C*: cost of executing verification; t_i: verification events; */V:* no verification; V_i: verification activity)

Without loss of generality, it is possible to assume that such verification strategy was determined optimal at the beginning of the program, that is, with the knowledge available at that point in time. Consider now that the verification activity V_1 at t_1 shows a tight margin with respect to the expected result of the activity. This may lead to a lower than expected confidence on the system being absent of errors that triggers the need for an additional, unplanned verification activity V_2 at t_1. Because the contract was fixed, such an activity needs to be contractually introduced through a change request.

Consider on the contrary, that the verification activity V_1 at t_3 showed much better results than previously expected. This may yield a higher than expected confidence on the system being absent of errors, potentially making verification activity V_2 at t_3 unnecessary or of little value, because of how confidence builds up on prior information (Salado & Kannan, 2018b; Salado, Kannan, & Farkhondehmaal, 2018).

Consider now the proposed set-based design approach, depicted on the bottom side of Figure 2. In this case, an optimal strategy is also determined at t_1. However, because the value of verification activities may change as results become available (Salado & Kannan, 2018b), a set (represented by the dotted lines connecting the dots) is considered instead of just one strategy, and only the first verification activity V_1 at t_1 is contracted at this point. This set is the set of all possible verification strategies that are consistent with the optimal verification strategy (that is, formed by all verification strategies that have the first activity in common).

Assume then that verification activity V_1 at t_1 provides low margin with respect to the expected results, as was the case before. With the updated confidence level, a new optimal strategy is selected within the remaining set. Then, the set is reduced to include only those verification activities that are consistent with the new optimal

strategy. In this way, verification activity V_2 at t_1 is contracted as well. The process of identifying new optimal strategies based on updated confidence and reducing the set of remaining verification activities to those consistent with the new optimal strategy, continues at each t.

Assume later in the system development that, as was the case when describing the current paradigm, verification activity V_1 at t_3 shows ample margin with respect to the expected result. The next assessment of the remaining optimal path yields a set of verification strategies that do not include verification activity V_2 at t_3. Based on this result, V_2 is not contracted at t_3. Consequently, this approach does not waste resources in activities that become no longer needed as verification evidence becomes available.

Process

The basic process that has been developed in this project to apply set-based design to the design of verification strategies consists of the following steps (Xu & Salado, 2019):

Step 1. Determine optimal verification strategy at Time 1.

Step 2. Choose first (timewise) verification activity (or subset of verification activities).

Step 3. Execute activity and update Bayesian network.

Step 4. Determine optimal remaining verification strategy and return to Step 2.

After each selection of an optimal strategy, the set of potential verification strategies is given by those strategies that share the first (timewise) verification activity (or subset of verification activities). Therefore, as the optimal remaining verification strategies are determined, the set shrinks until verification is completed.

In addition, the set of verification strategies can be further reduced by eliminating those sets that are dominated by optimal strategies throughout the system development. This reduction is useful for managing the resulting complexity.

An example of the evolution of the set of verification strategies after applying set-based design is provided in (Xu & Salado, 2019) and shown in Figure 3. At T_1, the optimal verification strategy contains V_1 at T_1. Two results are considered, either the activity *passes* or *fails*. In each case, the optimal strategy out of the set of remaining strategies can be computed. In both cases, the optimal strategy contains V_2 at T_2. The process continues by assessing how the optimal strategy changes on each path as the results of the next verification activity (in this case V_2 in each path) are known. This process is repeated until T_5. It should be noted how the result of each verification activity changes the optimality of the remaining verification strategy.

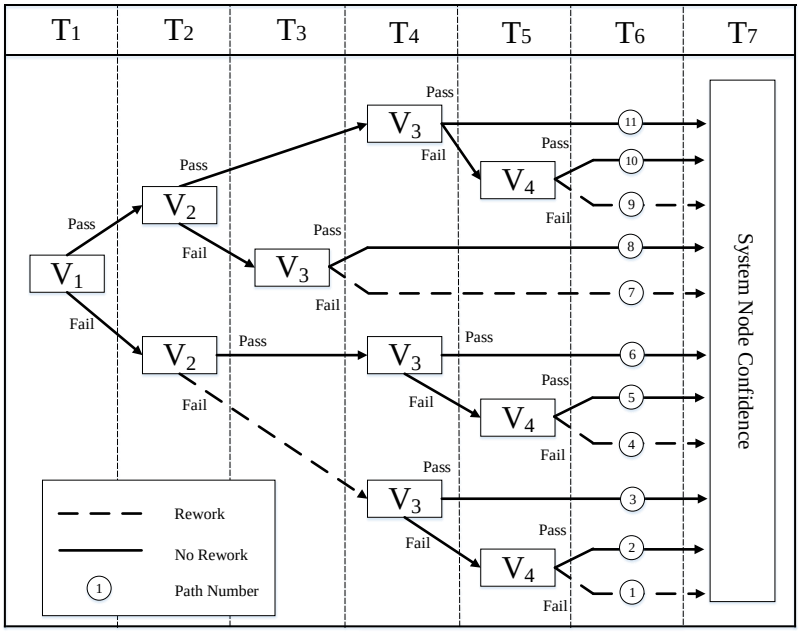

Figure 3. Verification path tree (Xu & Salado, 2019)

Overall in this example, eleven verification strategies dominate every other verification strategy in the set. Because of this, it suffices to work with an initial set of verification strategies (i.e., before T_1) that contains those eleven strategies. In case V_1 passes, the set shrinks to contain five strategies (strategies 7 to 11) after T_1 and before T_2. Otherwise, the set shrinks to contain six strategies (strategies 1 to 6). This process continuous until verification is completed. This evolution is consistent with the set-based design paradigm, since multiple alternatives are considered simultaneously and some of them are progressively discarded from the set until a single alternative finally remains.

Application

The proposed set-based approach was tested on four cases, which are reported in this section. The first case was used as a proof-of-concept. A simple verification network was employed, as a well as a simple model of rework activities. The second case employed a more sophisticated model of rework. The third and fourth cases were used as first attempts to address large verification networks. The third case leveraged the Monte Carlo method. The fourth case leveraged the parallel tempering algorithm.

Case 1. Proof-of-concept.

Note: This section has been slightly adapted from a publication by the authors prepared, submitted, and published during the period of performance of this research (Xu & Salado, 2019).

Description

The notional verification strategy in Figure 1 was used for this case. All nodes were assigned binary values for computational simplicity. This simplification does not affect the purpose of the case. System parameter nodes may take the values of *no error* or *error*, which are denoted by $\neg e$ and e, respectively. Verification activity nodes may take the values of *pass* or *fail*. A time vector $(T_1,...,T_n)$ is defined, where the element T_i precedes temporally the element T_{i+1} for each $i = 0,...,n-1$. No specific time unit was employed, because only temporal order is relevant to the example. Each element in the vector will be referred to as time event.

It was assumed that at most one verification activity is performed at each time event and that any given verification activity is performed at most once during the entire verification strategy. Furthermore, restrictions on the feasibility to perform a given verification activity at a given time event were defined and are listed in Table 1. The restrictions are intended to capture realistic constraints that may exist on the feasibility to perform a given verification activity at some point in the system

development. For example, it is likely that tests on prototypes can happen since an earlier time event than tests on the final product.

Table 1. Activity Constraint Table

Time event	Feasible verification activities
T_1	$L(T_1)=\{V_3,V_5\}$
T_2	$L(T_2)=\{V_3,V_4,V_5\}$
T_3	$L(T_3)=\{V_3,V_4,V_5\}$
T_4	$L(T_4)=\{V_5,V_6\}$
T_5	$L(T_5)=\{V_5,V_6\}$
T_6	$L(T_6)=\{V_5,V_6\}$

* $L(T_i)=\{\text{all feasible activities at } T_i\}$

The goodness or preference of a verification strategy was determined by three main factors: (1) its cost of execution, which is given by the fixed cost to execute each of its verification activities; (2) the expected cost to repair/rework the system when deemed necessary to do so as a function of the available verification evidence; and (3) the expected impact cost of the system exhibiting an error once deployed. Mathematically, the expected cost of a verification strategy S was modeled as:

$$E\left[C_T(S)\right]=\sum_{V\in\mathbf{V}}C_V(V)+\sum_{k=1}^{o}\sum_{j=1}^{n}\sum_{v\in L(T_j)}P(v)P(\theta_{jk}\mid v)\delta\left(\theta_{jk}\mid v\right)C_R\left(\theta_{jk}\right)+\sum_{k=1}^{o}\sum_{v\in\mathbf{V}^*}P(v)P\left(\theta_k=e\mid v\right)C_I\left(\theta_k=\mathrm{e}\right)$$

(0)

where:

$C_V(V)$ is the fixed cost to execute verification activity V ,

V Is the set of verification activities included in the verification strategy *S*,

v is a specific vector of verification results,

$P\left(\theta_{jk} \mid v\right)$ is the confidence level of the *k*th system parameter node at T_j given the verification results *v*,

$\delta(\theta_{jk} \mid v)$ is the indicator function that equals 1 if $P(\theta_{jk} \mid v) \leq H_l$, where H_l is a decision threshold, as will be explained in the next paragraph; otherwise its value is 0,

$C_R\left(\theta_{jk}\right)$ is the rework cost necessary to recover a failure detected during verification at T_j,

$\mathbf{V}^*$ is the set of verification results and rework efforts possible as per the previous rework decisions given the set of verification activities **V**,

$P\left(\theta = e \mid v\right)$ is the probability that the system exhibits an error, given the specific verification results *v*, and

$C_I\left(\theta = e\right)$ is the financial impact of the system exhibiting an error once it is operational.

The treatment of rework costs deserves additional discussion. A failed verification activity does not necessarily lead to rework; since rework is only necessary if worth doing. An automated rework decision process, caricaturized in Figure 4, is used in this case. Two confidence thresholds $\{H_l, H_u\} = \{0.4, 0.95\}$ distinguish between three decision zones, which are defined such that:

1) Zone 1 reflects a confidence state that is considered not acceptable. Therefore, if the confidence on the system being absent of errors drops to Zone 1, then a rework activity is executed. The rework activity results in the confidence increasing to the level it would be, had the verification activity yielded *pass* results. This is meaningful because the purpose of the verification activity that failed was to achieve certain confidence level.
2) Zone 2 reflects a confidence state that is in line with the confidence expected as the execution of the verification strategy progresses. Therefore, if (i) the confidence on the system being absent of errors falls in Zone 2 and (ii) the confidence level expected at completion of the verification strategy -assuming all remaining activities pass- falls in Zone 3, then the execution of the

verification strategy continues as planned. If this condition is not met, then a rework activity is planned until such an objective is reached.

3) Zone 3 reflects a confidence state that does not require the collection of additional knowledge; the engineer is *convinced* about the correct function of the system. Therefore, if the confidence on the system being absent of errors falls in Zone 3, rework activities are not executed. In addition, reaching Zone 3 implies for the set-based approach presented in this paper (with the corresponding dynamic contracting structure) that no other verification activity will be executed, and the system can be deployed. However, for the benchmark (with static contracting), it is assumed that remaining pre-contracted verification activities will still be executed.

Probability assignments use synthetic data and are given in the Appendix. Following the modeling approach presented in (Salado & Kannan, 2019), prior beliefs were assigned to system parameter nodes, which capture the initial belief on the state of the system (i.e., being absent of errors), and conditional probability tables were created for the verification activity nodes. Posterior beliefs were calculated for system parameters through Bayesian update of the outcomes of the verification activity nodes. Probability update was conducted in this study using the Bayesian Network Toolbox for MATLAB®, which estimates the posterior probabilities of all nodes by the variable elimination method.

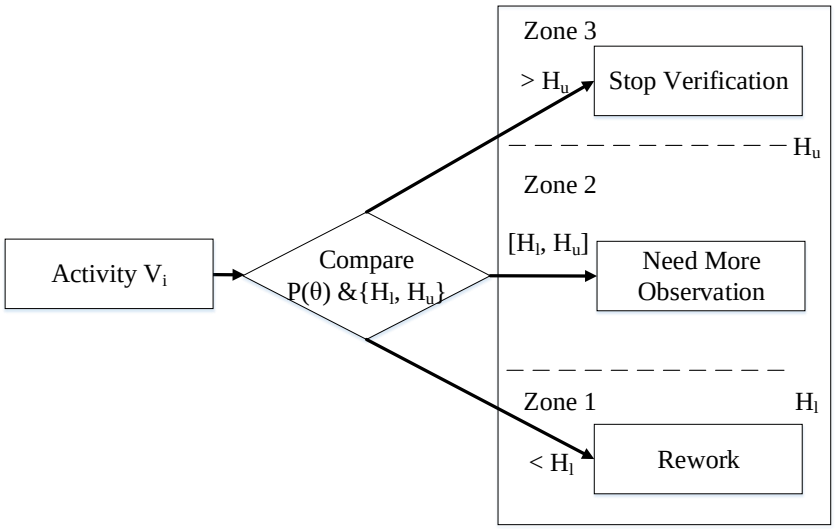

Figure 4. Zones for deciding next verification activity and need for rework (Xu & Salado, 2019).

Cost values employed in this case, given in the Appendix, were also synthetic, but reasonable. The following assumptions were made: (1) rework cost increases with time, (2) the impact cost during deployment is much larger than the rework cost and the verification cost; (3) rework cost is in general higher than verification execution cost; and (4) verification execution cost is positively related to the information it yields.

Results

Given the constraints in Table 1, an initial set of 198 verification strategies could be enumerated before the first time-interval. Among them, the optimal one is S_1 = (V1, V2, NoV, V3, V4, NoV), where NoV indicates that no verification activity is executed at that time interval. This strategy has an expected total cost of $3,226k and an initial confidence on the system being absent of errors of 0.76. As discussed, S_1 is used as the baseline verification strategy for the benchmark.

As an example, the evolution of one of the paths for the proposed set-based approach is described. V1 is executed in the first time-interval because it is part of the optimal strategy identified before initiating the execution of the verification strategy. If the verification activity passes, the number of verification strategies remaining in the set reduces to 55 (all strategies that begin with V1) and the confidence on the system being absent of error increases to 0.84 (as determined through Bayesian update of Figure 1). The optimal verification strategy out of the remaining set becomes S_2= (V1, V2, V3, V4, NoV, NoV), with a lower expected cost of $2,994k. On the other hand, if the activity fails, the set of remaining verification activities would contain 115 elements and the confidence on the system being absent of error would drop to 0.57. Since this level is still larger than 0.40, the rework activity would not be entertained yet. The process repeats again by identifying a new optimal strategy and reducing the set accordingly until the verification activity on the last time interval is executed.

The possible set reductions led to 11 feasible paths for the proposed set-based approach. As illustrated in Figure 3, the set of all possible paths could be represented as a tree plot. The expected cost of each approach to design verification strategies was calculated as the sum of the cost of each path weighted

by its resulting probability of occurrence. The probability of occurrence for each path was computed as the product of all the probabilities of all activities along the branch. Detailed results are shown in Table 2.

Table 2. Table of All Cost Items at T_7

Path Number	Path Probability (*PP*)	$P(\theta_2 = \neg\text{error})$	E[C_I]	C_R	C_V	Path Cost ($CP=E[C_I]+C_R+C_V$)
1	0.0295	0.9077	5538	700	425	6663
2	0.0116	0.9077	5538	200	425	6163
3	0.1169	0.9657	2058	200	350	2608
4	0.0265	0.9077	5538	500	425	6463
5	0.0104	0.9077	5538	0	425	5963
6	0.1051	0.9657	2058	0	350	2408
7	0.0446	0.9364	3816	300	225	4341
8	0.0554	0.9364	3816	0	225	4041
9	0.0936	0.9316	4104	500	425	5029
10	0.0449	0.9316	4104	0	425	4529
11	0.4615	0.9750	1500	0	350	1850

Similarly, the benchmark could yield 16 possible paths. All paths are shown in Figure 5 (dotted, red lines represent benchmark paths; solid, blue lines represent set-based paths). The vertical axis represents the total expected cost of the verification strategy on each time interval. The resulting cost is given therefore after completion of the last time interval (to the right extreme in the plot). The total expected cost of the set-based approach is $\sum_{i=1}^{11} PP_i \times PC_i = \$3{,}004\text{k}$, which is smaller than that of the benchmark, $3,214k. This result provides an indication that the proposed approach yields indeed more valuable verification strategies than the benchmark, although additional cases need to be run to confirm this result.

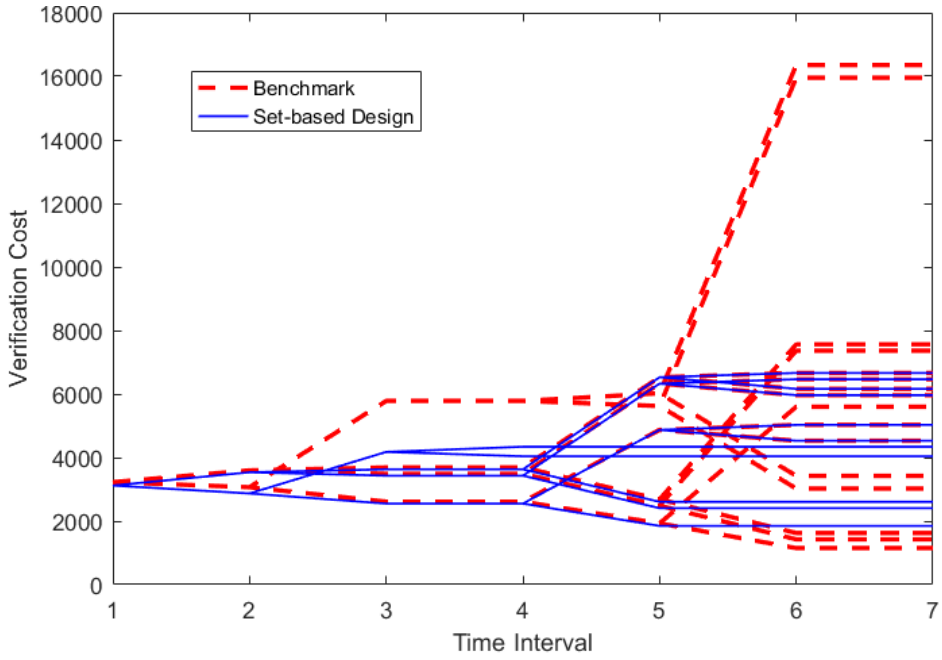

Figure 5. Comparison between Set-based Design and Traditional Strategy (Xu & Salado, 2019)

Figure 5 provides in addition an interest insight about the properties of the proposed set-based approach to design verification strategies and contract verification activities. As can be seen, the amplitude of the tree corresponding to the benchmark approach (red dotted line) is larger than that of the set-based design method (blue solid line). This indicates that the benchmark approach responds more slowly to adjusting its parameters than the set-based design approach when receiving information from verification evidence. In cost control terms, this indicates that the benchmark approach is inefficient when compared against the proposed set-based approach.

Case 2. Improved model of rework

Description

Consider the simple overarching verification network in Figure 6. It represents the way in which a set of available verification activities provide information about a system parameter θ_S (e.g., the mass of the system). In the figure, θ_C represents another parameter that provides information about θ_S (e.g., the mass of a system

component), V_1 is a verification activity that provides information about θ_C (e.g., a test of the mass of a system component), and V_2 is a verification activity that provides information about θ_S (e.g., a test of the mass of the system).

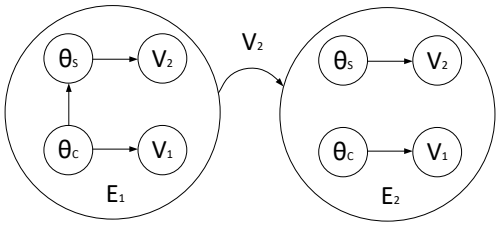

Figure 6. Overarching verification network

Five verification strategies can be devised by leveraging the overarching network (notation from (Salado & Kannan, 2018a) is used):

$S_1 = (\varnothing, \varnothing)$

$S_2 = (\{V_1\}, \varnothing)$

$S_3 = (\{V_2\}, \varnothing)$

$S_4 = (\{V_1, V_2\}, \{(V_1, V_2)\})$

$S_5 = (\{V_1, V_2\}, \{(V_2, V_1)\})$

It was assumed that S_5 is not meaningful and therefore it was not further considered.

The cost to execute a verification activity is denoted by σ_V. Table 3 lists the cost to execute each verification strategy. It was assumed that no overlap exists in the cost of executing the verification activities.

Table 3. Cost to execute verification strategies

Strategy	Cost function
S_1	$\sigma_V(S_1) = \$0$
S_2	$\sigma_V(S_2) = \sigma_V(V_1) = \$200\text{K}$
S_3	$\sigma_V(S_3) = \sigma_V(V_2) = \$200\text{K}$
S_4	$\sigma_V(S_4) = \sigma_V(V_1) + \sigma_V(V_2)$

The cost impact associated to deploying the system with an error is denoted by σ_I. Table 4 lists the expected costs of impact for each strategy. It was assumed that $\sigma_I = 10,000\text{K}$.

Table 4. Impact cost of deploying the system with an error

Strategy	Cost function
S_1	$E\left[\sigma_I(S_1)\right] = P(\theta_S = e)\cdot\sigma_I$
S_2	$E\left[\sigma_I(S_2)\right] = P(\theta_S = e \mid V_1 = p)\cdot\sigma_I$
S_3	$E\left[\sigma_I(S_3)\right] = P(\theta_S = e \mid V_2 = p)\cdot\sigma_I$
S_4	$E\left[\sigma_I(S_4)\right] = P(\theta_S = e \mid V_1 = p, V_2 = p)\cdot\sigma_I$

Note that $E\left[\sigma_I(S_3)\right] = E\left[\sigma_I(S_4)\right]$ because V_1 becomes disconnected from θ_S once V_2 is known.

Rework cost is denoted by σ_R. The key aspect is that the cost of rework depends on when the rework happens or, more accurately, on whether rework requires integration and de-integration activities or not. Hence, it is necessary to capture the cause of the error, as well as the moment in which the error is found. It was assumed that rework results in a state of knowledge equivalent to $V = p$. This is because, in the theoretical framework used in this research, system attributes are not accessible, but only verification evidence is (Salado & Kannan, 2019).

Contrary to the previous case, it was assumed in this case that rework is performed as soon as a verification activity fails. This implies the following:

- For S_1, $E\left[\sigma_R(S_1)\right] = 0$ because, since there is no verification activity executed, errors cannot be found and rework activities initiated.
- For S_2, $E\left[\sigma_R(S_2)\right] = P(V_1 = \neg p)\cdot\sigma_R(C,C)$, where $\sigma_R(A,B)$ indicates that rework happens for assembly *A* when integrated at assembly level *B*. In this case, (C,C) means that rework happens on the component when it is at the component level (that is, when the component is not integrated at system

level). Only $\sigma_R(C,C)$ is considered in the model because, since no verification at system level occurs, errors can only be found at the component level.

Calculation for S_3 becomes more sophisticated because, while the failure is detected on a verification activity at the system level, the error may result from an error at system level and/or an error at component level (note that in some cases solving the problem at the component level automatically solves the problem at the system level and in some cases the system level problem persists and also needs to be fixed). This needs to be considered in the calculation of the expected rework cost. The following basic algorithm was used:

1) If an error is found, try to solve at system level.
2) If not solvable, try also at component level.

Note that a different algorithm could have been defined, trying to fix the problem at component level before trying at system level. However, based on experience, it was assumed that de-integration activities are less preferred. Under these conditions, the expected rework cost for S_3 is given by Eq. (3),

$$E\left[\sigma_R(S_3)\right] = P(V_2 = f)\cdot\left[\sigma_R(S,S) + P(\theta_S = e, \theta_C = e \mid V_2 = f)\cdot\sigma_R(C,S)\right] \qquad (3)$$

The following aspect is of interest in the previous equation. Note that, if the verification activity fails, rework automatically happens at the system level. As stated, rework at the component level is performed only if the problem persists. This was modeled by the probability that there is an error at both the system level and the component level. This is because:

1) If the error was only at the system level, then the rework at system level would fix it.
2) If the error was only at the component level, then there is not really a problem at system level and the fix would also work.
3) The cost of rework of system level is already accounted for, so this is why only the cost of the component level fixed is considered in that case.

Calculation for S_4 builds upon the same idea:

1) If the component level verification activity fails, then a rework activity at the component level occurs. Afterwards, if the system level verification activity fails, the same situation as in S_2 applies with the difference that probability of errors is conditioned to the component level activity passed (because of the rework activity).
2) If the component level verification activity passes and then the system level verification activity fails, the same situation as in S_3 applies with the difference that probability of errors is conditioned to the component level activity passed.

Under these conditions, the expected rework cost for S_4 is given by Eq. (4),

$$E\left[\sigma_R\left(S_4\right)\right]=P\left(V_1=f\right)\cdot\left(\sigma_R\left(C,C\right)+P\left(V_2=f\,|V_1=p\right)\cdot\sigma_R\left(S,S\right)\right)+P\left(V_1=p\right)\cdot P\left(V_2=f\,|V_1=p\right)\cdot \left(\sigma_R\left(S,S\right)+P\left(\theta_S=e,\theta_C=e|V_2=f,V_1=p\right)\cdot\sigma_R\left(C,S\right)\right)$$

(4)

Table 5 lists the corresponding rework cost used in the model.

Table 5. Rework costs

$\sigma_R(x,y)$		y	
		C	**S**
x	**C**	$200K	$1,000K
	S	n/a	$500K

Note that all cost figures are synthetic. Probability assignments use synthetic data and are given in the Appendix. As in the previous case, the modeling approach presented in (Salado & Kannan, 2019) was employed. Prior beliefs were assigned to system parameter nodes, which capture the initial belief on the state of the system (i.e., being absent of errors), and conditional probability tables were created for the verification activity nodes. Posterior beliefs were calculated for system parameters through Bayesian update of the outcomes of the verification activity nodes. Probability update was conducted in this study using the Bayesian Network Toolbox for MATLAB®, which estimates the posterior probabilities of all nodes by the variable elimination method.

Results

Because of the size of the network and the input data, this case is not able to distinguish between the current acquisition paradigm and set-based design. However, the case was only used to explore the application of the refined rework model, so the case is still useful.

Results are shown in Figure 7. Two time-events are represented, one at Time Interval = 1 (denoted by T_1) and one at Time Interval = 2 (denoted by T_2). Verification activities V_1 and V_2 are conducted at T_1 and T_2, respectively. Solid continuous lines are used for visualization purposes. Bifurcations differentiate the cost of potential paths should the verification activity pass or fail. Because of the setup of the case, the cost differences are caused only by the rework actions. The paths with positive slope indicate that the verification activity failed and, consequently, a rework activity was initiated. On the contrary, the paths with negative slope indicate that the verification activity passed and, consequently, rework activity was not initiated. The key insight of the picture is the consistency with which rework at different levels of integration is treated; in line with the input data. As can be seen, the delta rework cost after V_2 is larger than after V_1. This is, as discussed, because not only rework at higher integration levels is more expensive, but there is a chance that the problem at system level is caused by a problem at component level. Such de-integration effort considerable increases the resulting rework cost.

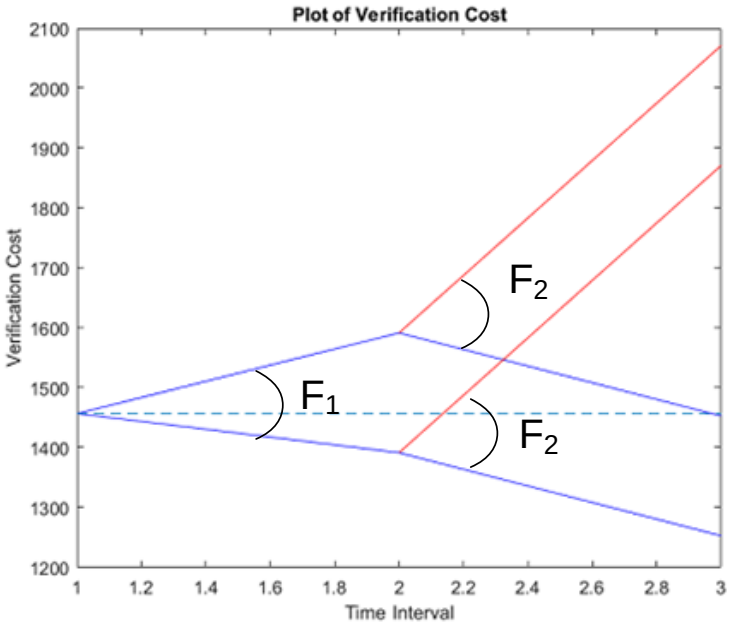

Figure 7. Plot of Verification Paths

Case 3. Extending the size of the verification strategy: A Monte Carlo approach

Description

The partial verification network of a space system in Figure 8 was used for this case. It captures all verification activities that may be leveraged to verify three system parameters: field of view (denoted by θ_1), modular transfer function (denoted by θ_3), and mechanical load (denoted by θ_2). The space instrument consists of a telescope, a spectrometer, and camera. The network also includes telescope, spectrometer, and camera parameters that provide information about the three system parameters to be verified. Furthermore, the network includes various verification models that may be used to support verification activities (e.g., mathematical models, prototypes, and the final product). These parameters are denoted by θ_i, with $i \neq 1,2,3$. Verification activities are noted by V_i. The details of the verification network are given in (Salado & Kannan, 2019). Overall, the network has 21 parameter nodes and 29 verification activity nodes. The network was characterized with notional values. A partial snapshot is shown in the Appendix for

information purposes (note: the full dataset is too large for representation in this report).

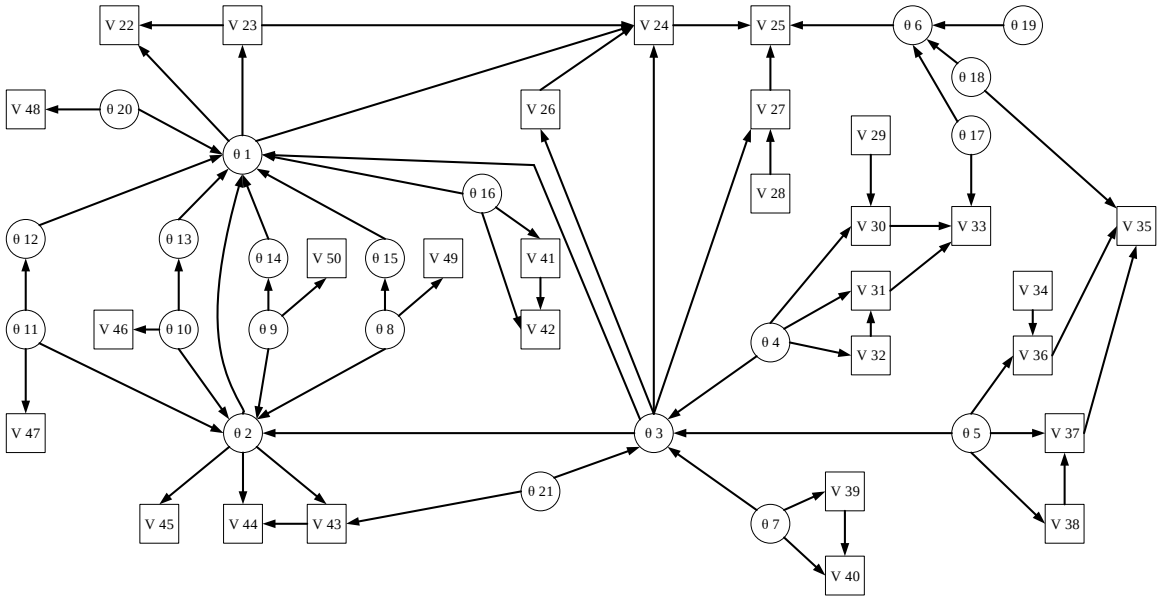

Figure 8. Verification network used in Cases 3 and 4 (Salado & Kannan, 2019)

Method

The Monte-Carlo (MC) method has been explored to cope with the complexity resulting from large verification strategies. This is necessary because brute force (i.e., exhaustively evaluating all possible verification strategies for a system) is infeasible in realistic systems. The explored MC method has two stages: importance sampling (Stage I) and importance resampling (Stage II). At Stage I, the expected cost of each candidate node is obtained, forming a cost vector. The task of importance sampling is to obtain one convergent cost vector. That is, after Stage I, the cost vector is no longer changed and is used to calculate the importance of each node. At Stage II, another set of sample paths are generated. They are used to take a close observation at those nodes with the larger importance identified in Stage I. The target of this stage is to identify the optimal node with the lowest expected cost. The basic idea is to compare the top two candidate nodes by the two-sample *t*-test. The overall procedure of this method is shown in Figure 9. It consists of three loops.

They are designed for cost vector, node importance, and required sample sizes separately. The final decision is made according to the *t*-test.

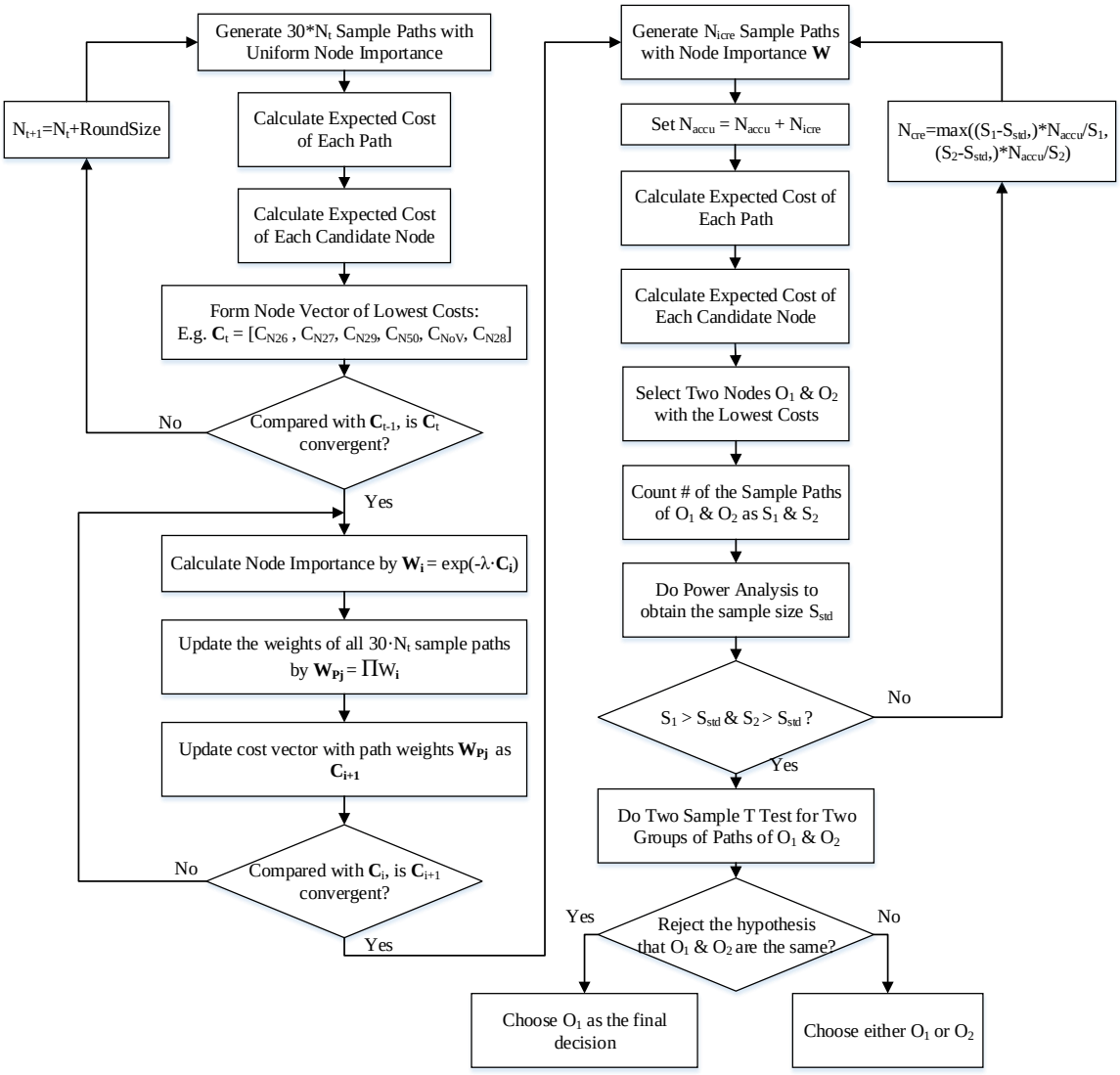

Figure 9. MC method process description

In the first loop, a convergent cost vector is obtained. To ensure all nodes have enough samples for observations, the first node of each path is fixed and the same number of samples N_t is allocated for each node available at the first time-interval. The total number in this case was 30* N_t. Criterion of convergence was determined as follows. First, the cost vectors are calculated round by round. The

round size between two sequential rounds is constant. Then two cost vectors are compared. If they are close or similar enough with each other, this loop ends. If the number of nodes that appear in both cost vectors is larger than one threshold, it is claimed as convergent.

In the second loop, the task is to determine node importance. The motivation of this step is to assign more weights to those important nodes in the following resampling process. As low-cost nodes are preferable in this case, the exponential function $W = \exp(-\lambda \cdot C)$ has been adopted. The adjustment factor, λ, is used to control the shape of the mapping function. If λ is too large, those nodes with large costs may not have enough samples for observation. But if λ is too small, the following resampling process may generate more samples to guarantee the performance. In order to make λ meaningful, λ is set such that the maximum and minimum costs of the sample paths satisfy $\frac{\exp(-\lambda \cdot C_{\max})}{\exp(-\lambda \cdot C_{\min})} = 10$.

In the third and final loop, importance resampling is performed to compare the costs of two nodes that are the most likely to be chosen. Since the method relies on the technique of hypothesis test, a power analysis is done first to ensure there are enough samples for testing. Hence, the MC method is conducted incrementally. The samples are generated batch by batch until the power analysis passes. At each round, the two nodes O_1 & O_2 of the minimum costs are chosen. The costs of all paths starting with O_1 & O_2 are collected as two distributions. In this case, for simplicity and without loss of generality, it is assumed that they follow normal distributions. First, their means and standard variances, which are denoted as M_1, M_2, SD_1, and SD_2, are calculated. Then, the error $M_1 - M_2$ is used as the estimated difference and $\sqrt{\frac{SD_1^2 + SD_2^2}{2}}$ as the estimated standard deviation σ. They are used to derive the required sample size S_{std}. The difference between S_{std} and S_1(or S_2) is used to determine the extra sample size of the next round. Notably, all previous samples would be reused until the power analysis passes. In this way, the conditions of the hypothesis test can be satisfied. Finally, the two-sample *t* test is implemented

for the null hypothesis that two nodes O_1 & O_2 share the same distribution. If the hypothesis is rejected, then O_1 is significantly better than O_2. Otherwise, there is no difference between O_1 & O_2 and either of them may be chosen as the final decision.

Results

When applying the MC approach to the case in Figure 8, the first loop stopped when 17 sample paths for each candidate node were evaluated, yielding a total of 510 paths. As examples, the cost vector for N_t=17 is [26, 27, NoV, 30, 46, 28] and the cost vector for N_t=14 is [26, 27, NoV, 46, 30, 41]. They both share 26, 27, NoV, 46, 30. So it is judged as convergent. Their mean values are shown in Tables 6 and 7 for information.

Table 6. Means when Nt=17

No V	*N 22*	*N 23*	*N 24*	*N 25*	*N 26*	*N 27*	*N 28*	*N 29*	*N 30*
11038	14811	13449	15603	14481	**9746**	10269	12202	12406	12046
N 31	*N 32*	*N 33*	*N 34*	*N 35*	*N 36*	*N 37*	*N 38*	*N 39*	*N 40*
14058	13072	15199	12936	15523	12546	15393	13149	14110	16120
N 41	*N 42*	*N 43*	*N 44*	*N 45*	*N 46*	*N 47*	*N 48*	*N 49*	*N 50*
12954	13444	13158	14994	14109	12098	13285	12488	13047	12570

Table 7. Means when Nt=14

No V	*N 22*	*N 23*	*N 24*	*N 25*	*N 26*	*N 27*	*N 28*	*N 29*	*N 30*
11478	15037	14432	15465	14910	**9857**	10691	12983	13509	12118
N 31	*N 32*	*N 33*	*N 34*	*N 35*	*N 36*	*N 37*	*N 38*	*N 39*	*N 40*
13063	13505	14629	12661	15822	13379	14546	14292	12725	15463
N 41	*N 42*	*N 43*	*N 44*	*N 45*	*N 46*	*N 47*	*N 48*	*N 49*	*N 50*
12311	13098	14249	15050	13491	11920	15095	12828	12970	13131

Setting in the second loop the ratio to 10, the resulting node importance after normalization is shown for information in Table 8.

Table 8. Node Importance when λ=3.97

No V	*N 22*	*N 23*	*N 24*	*N 25*	*N 26*	*N 27*	*N 28*	*N 29*	*N 30*
0.0471	0.0265	0.0325	0.0212	0.027	**0.0557**	0.0524	0.0396	0.0379	0.04
N 31	*N 32*	*N 33*	*N 34*	*N 35*	*N 36*	*N 37*	*N 38*	*N 39*	*N 40*
0.0286	0.0349	0.0239	0.0352	0.0233	0.04	0.0229	0.0326	0.0291	0.0209
N 41	*N 42*	*N 43*	*N 44*	*N 45*	*N 46*	*N 47*	*N 48*	*N 49*	*N 50*
0.0348	0.032	0.0337	0.0251	0.0272	0.038	0.0328	0.0372	0.0325	0.0355

For the Third Loop, the node importance is fixed as those in Table 8. The loop stopped when there were 2200 samples in total. O_1 & O_2 are N 26 and N 27. Their sample sizes were 129 and 85. Their means and standard deviations were O_1 [9436, 1765.1], O_2 [10240, 1694.3]. The required sample size was 77. Because 77 < 85 and 77< 129, the *t* test could be done. The p value was 0.0006. Hence, the null hypothesis was rejected and declare that O_1 is better. Therefore, at T_1, Node 26 should be verified first.

The performance of this method to find optimal verification strategies within a set-based design framework are shown in the Findings section.

Case 4. Extending the size of the verification strategy: A state-based approach

Description

The case in Figure 8 is used in this case as well. A different rework model was used in this case. It was assumed that rework is always triggered when the confidence drops below a preset threshold. Revenue was also incorporated as a factor, as opposed to cost of impact of error. Revenue was modeled as positive if a present threshold of confidence was achieved (that is, the system is deployed), and zero if the threshold would not be reached. Mathematically, the expected overall cost of one completed verification strategy was modeled as:

$$C_{total} = \sum_i C_{V_i}(V_i) + \sum_i C_{R_i}(t)\delta(\theta_k < \mathrm{H}_1|\boldsymbol{V}) - \sum_k R_{E_k} Pr(\theta_k|\boldsymbol{V})\delta(\theta_k < \mathrm{H}_2|\boldsymbol{V})$$

where $C_{V_i}(V_i)$ is the fixed cost to execute verification activity V_i, C_{R_t} is the rework cost at each time point *t*, and $P(\theta_k|\boldsymbol{V})$ is the confidence level of the *k*th system parameter node given the verification results $\boldsymbol{V}$. The term $\delta(\theta_k|\boldsymbol{V})$ acts as an indicator function that equals 1 if $P(\theta_k|\boldsymbol{V})$ is smaller than some lower threshold; otherwise its value is 0. The lower threshold for rework is denoted by H_1 and the upper threshold for deployment is denoted by H_2.

All CPTs were generated according to the Noisy-OR model. The dataset is too large and has not been included in this report. The revenue coefficient was set to \$100,000. H_2 was set to 0.9 and H_1 to 0.5. Five time-intervals were defined. Verification activity costs and basic rework costs are provided in the Appendix. Rework cost was modeled as being exponentially proportional to the time interval when the rework happens (Blanchard & Fabrycky, 1990). Specifically, the factors [1, 1.41, 2, 2.82, 4] were applied to each time-interval respectively.

Method

In this case, the verification strategy design problem is treated as a kind of Markov Decision Process (MDP). The highly structured state space of the verification strategy imposes however challenges to computational efficiency. Parallel Tempering (PT) (Earl & Deem, 2005), which is also known as replica exchange Markov Chain Monte Carlo (MCMC) method and is an extension of the traditional Simulated Annealing (SA) method, is employed. The PT method contains N replicas of the system that are simulated simultaneously at various settings. Although this requires computing several replicas instead of a single system, it still improves the overall efficiency with respect to the traditional Monte Carlo method employed in Case 3. The PT algorithm that was employed in this case is shown in Table 9. Selection of number of swaps, denoted by N_s, and number of MC iterations, denoted by N_{it}, was done following the guidelines in (Wang, Hyman, Percus, & Caflisch, 2009).

Table 9. PT algorithm employed in Case 4

Input: $\{T_k\}$ —— Temperature set; M —— Number of temperatures;

N_{it}—— Number of MC iterations between swaps; N_s —— Total number of swaps

Output: the optimal sample x_{opt}

1. Initialize all M replicas $\{X_m\}$
2. For i = 1 … Ns
3. For m = 1 … M
4. Run Metropolis method for all M replicas $\{X_m\}$ for N_{it} iterations.
5. End
6. Generate one random number k from [1, 2, … , M-1].
7. Swap X_k with X_{k+1} with probability $p = \min(1, \exp(\Delta\beta\Delta \mathrm{E}))$.
8. End
9. Choose the optimal sample x_{opt} from all M replicas
10. Set $\{X_m\}$ as the initial ones and go to Step 2
11. Stop if x_{opt} remains the same

The parameters N_{it} and N_s were set as 10 as 20, respectively, to accelerate the computation efficiency. Their suitability for convergence was confirmed by leveraging Step 11 in Table 9. The temperature vector was designed as [15.6 31.3 62.5 125 250 500 1000 2000 4000 8000 16000 32000 64000]. This vector derives from the interval of the overall verification strategy cost, which leads to the ranges $\Delta \mathrm{E}$ = [-1.4*10^{-5}, 1.4*10^{-5}] and $\Delta\beta$ = [2.14*10^{-5}, 0.03]. Finally, a geometric progression of temperatures satisfying $T_{i+1} / T_i = 2$ was assumed.

Two approaches were designed to update the path sample. In the first one, the path is updated by randomly switching two activities of each path. In the second one, a randomly selected activity is updated. Importance weights are used to select the activity to be changed. The importance weight of one activity V_i was calculated by observing the response range $|P(V_i \mid V_s = T) - P(V_i \mid V_s = F)|$ when the selected

activity is assigned as P/F. During the actual update, the first approach is used with 80% likelihood while the second one with 20% likelihood. In this way, one new path can be generated without significantly modifying the original one.

Results

To illustrate the PT algorithm, the optimization process at the first time-interval, that is, for the state where no activity has been done yet, is shown. The configuration of all 13 temperatures is shown in Figure 10. When there is one swap between two neighboring temperatures, the two corresponding dash lines will intersect with each other. In particular, the configuration that results in the optimal path is plotted with a red solid line. It is generated at the track of Temperature 500 and decreases to the lowest temperature all the way. Their corresponding real-time overall costs and global optimal costs are shown in Figures 11 and 12, respectively. The real-time and global optimal costs of the optimal configuration are also plotted as red solid lines. Stability of the optimal configuration was achieved after 600 iterations. The determined optimal path was [V_{26}, V_{24}, V_{22}, V_{34}, V_{46}], so verification activity V_{26} is the appropriate action for the next time-interval and the set of possible verification strategies shrinks to include only those that begin with such activity. All other verification strategies are discarded.

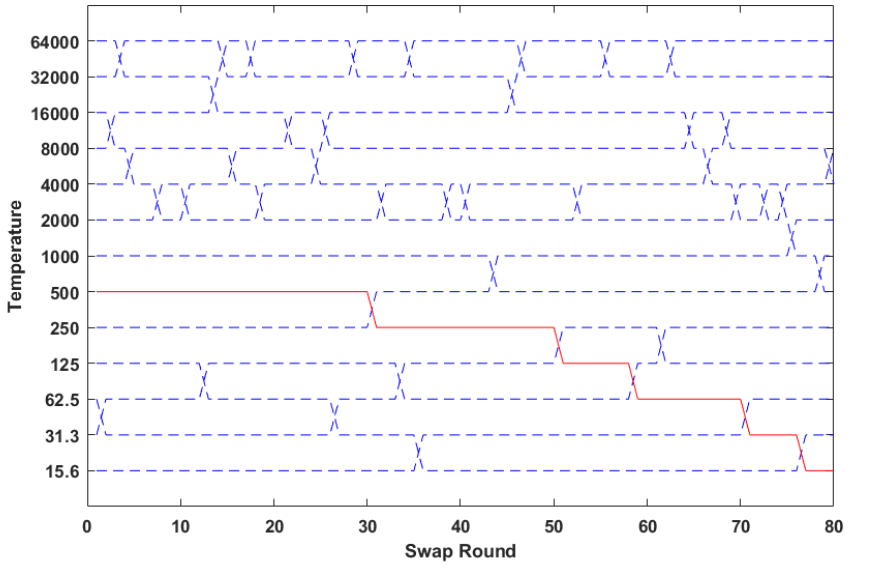

Figure 10. Swap plot of the configurations of all temperatures

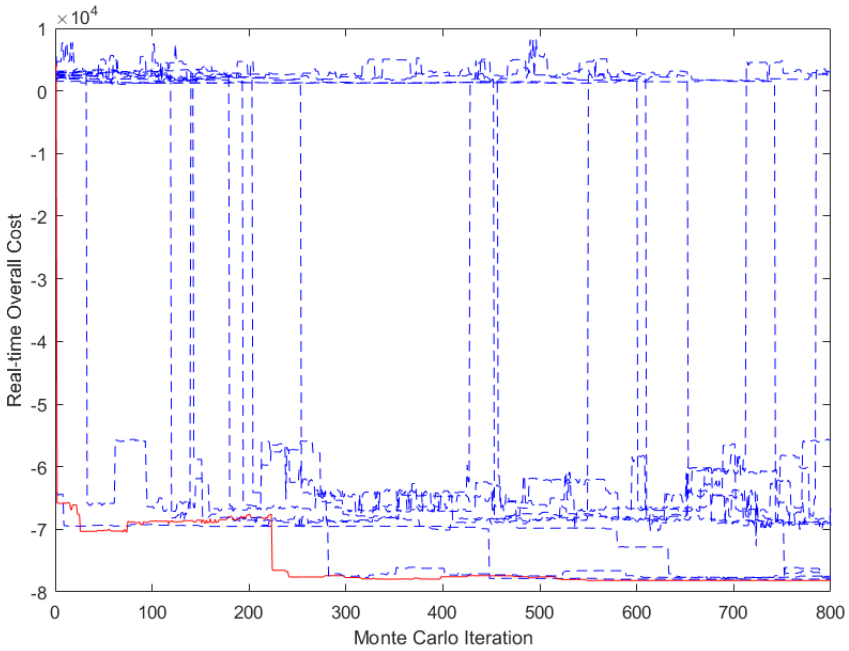

Figure 11. Overall cost of the configuration that generates the optimal path

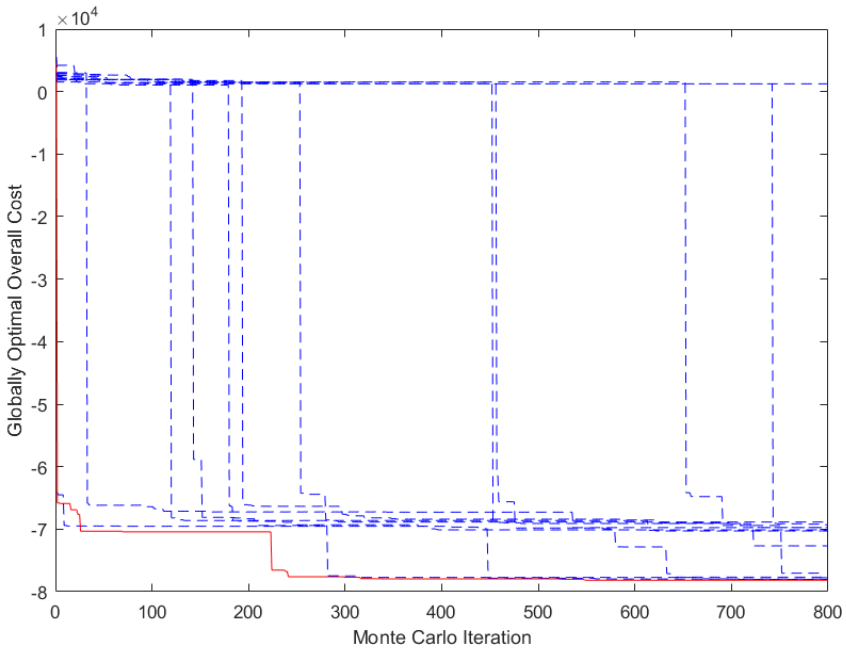

Figure 12. Globally optimal overall cost of all configuration

This method is then recursively applied on each time-interval, where a new optimal verification strategy is found, its first activity is then selected for action and

the set of possible verification strategies shrunk accordingly, and based on the results of this newly selected verification activity, a new remaining verification strategy is determined. The results are summarized in Table 10. For each state, the optimal activity for the next time interval is shown in bold and the expected costs of the optimal paths are listed in the fourth column. As can be seen, the verification strategy adapts as necessary, as a function of the results of the different verification activities.

Table 10. The state-action table

Time Interval	State (S_i)	Optimal Path (P_i)	Expected Cost (C_{Pi})	Note
T_1	Null	[**26**, 24, 22, 34, 46]	-78173	
T_2	V_{26} = F	[26, **22**, 48, 43, 23]	-69696	
T_3	V_{26} = F, V_{22} = F/T	[26, 22, **23**, 48, 44]	-61537	Rework if V_{22} = F
T_4	V_{26} = F, V_{22} = T, V_{23} = F	[26, 22, 23, **Null**, Null]	3995.6	
T_4	V_{26} = F, V_{22} = T, V_{23} = T	[26, 22, 23, **48**, 43]	-76865	
T_5	V_{26} = F, V_{22} = T, V_{23} = T, V_{48} = F	[26, 22, 23, 48, **Null**]	4395.6	
T_5	V_{26} = F, V_{22} = T, V_{23} = T, V_{48} = T	[26, 22, 23, 48, **43**]	-79553	
T_2	V_{26} = T	[26, **24**, 46, 22, 0]	-82347	
T_3	V_{26} = T, V_{24} = F/T	[26, 24, **22**, Null, Null]	-79764	Rework if V_{24} = F
T_4	V_{26} = T, V_{24} = T, V_{22} = F	[26, 24, 22, **Null**, Null]	4061.3	
T_4	V_{26} = T, V_{24} = T, V_{22} = T	[26, 24, 22, **Null**, Null]	-88094	

These results are also graphically shown as a tree plot in Figure 13. The posterior possibility of each result accompanies each branch of the activity node for information. All these possible activities constitute the final policy for the verification strategy of the system under study. The final path out of the 12 possible will depend on how each verification activity materializes, that is, on the results of each verification activity as they are executed at different time-intervals.

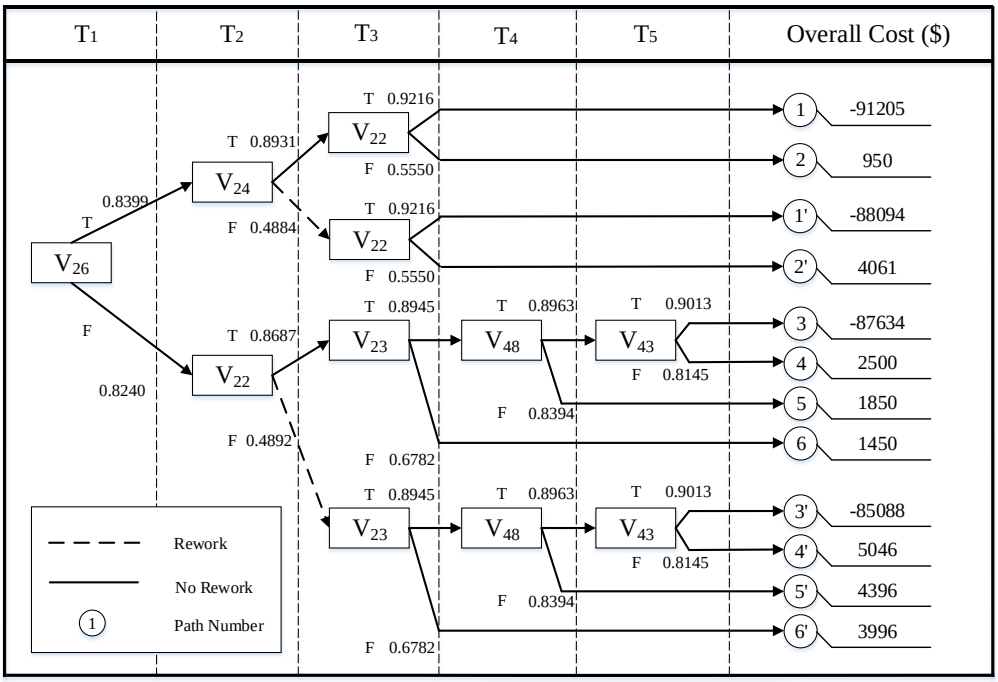

Figure 13. Plot of all possible activities

The performance of this method to find optimal verification strategies within a set-based design framework are shown in the Findings section.

Findings

The previous four cases provide two main results. First, the set-based design approach to dynamically design and contract verification strategies yields verification strategies with higher expected value than those designed using current practice. Consider the red (solid) and blue (dashed) lines in Figure 14, which plots the real-time verification costs for various verification paths in the context of the verification problem defined in Cases 3 and 4 for the network depicted earlier in Figure 8. The blue (dashed) line represents the optimal verification strategy designed using the current paradigm to contracting verification strategies. As discussed previously, this paradigm consists of determining a verification activity at the beginning of the project and execute it regardless of the results, with the caveat that additional verification activities may be contracted if the achieved confidence is not sufficiently high. The red (solid) line represents the optimal verification strategy designed applying set-based design, as developed in this research project. Parallel tempering was employed to determine the optimal verification strategies under both design paradigms in order to factor out the approximation method. Whereas the expected value of the optimal verification strategy under the current paradigm (static verification strategy) was $78,173, the expected value of the verification strategy under set-based design (dynamic verification strategy) was $80,980. (Note: value has been determined as the inverse of cost).

Second, parallel tempering provided better results than Monte Carlo to deal with the complexity of large networks. The results obtained with Monte Carlo method, when applied to the current paradigm (static verification) are plotted as a green (dashed dotted) line in Figure 14. The best verification strategy identified using the Monte Carlo method had an expected value of $60,291, well below the one identified using parallel tempering, as described above.

It should be noted that all values that have been used in these simulations are synthetic. Specific results should not be interpreted as a metric of performance

improvement. Instead, the results provide an indication of the potential value of the method developed in this research project.

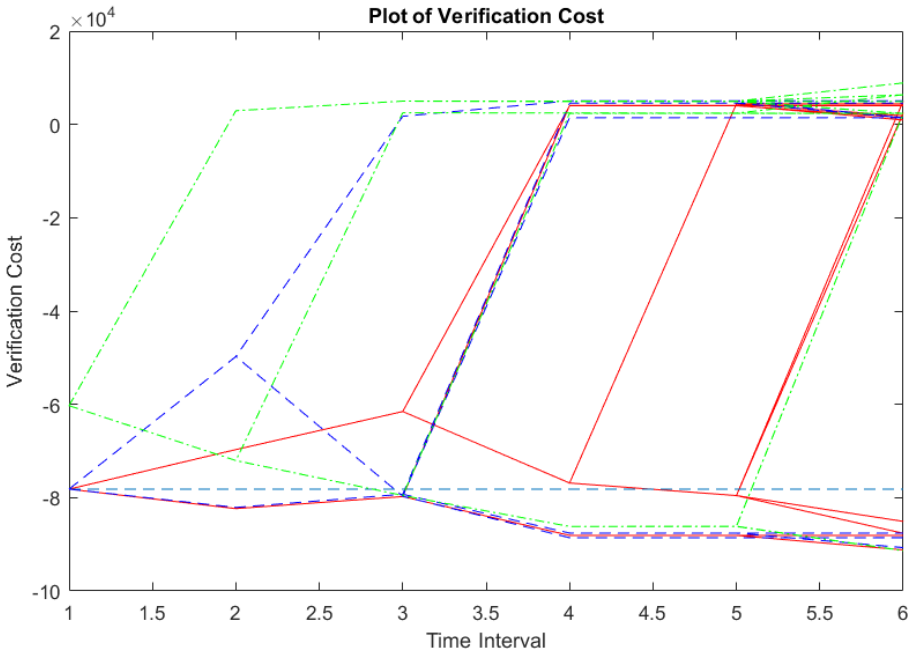

Figure 14. Comparison of Three Strategies

Recommendations and Future Directions

This research has shown that there is value in not anchoring to a specific verification strategy early in a system development, specially for system with long development cycles. Dynamically adapting the verification strategy as results become available is necessary to effectively and efficiently leverage the knowledge gained through the execution of verification activities. Not doing so is inherently suboptimal, leading to spending resources in unnecessary verification activities and creating gaps in the verification coverage, with its corresponding risk of failure or degraded performance once the system is deployed and put into operation.

This work suggests that transforming the way in which verification activities are planned and contracted in acquisition programs is essential to improve system affordability and mission early success. Government should not contract a fixed verification strategy at the beginning of a project, but rather allocate a verification budget in a similar way in which contingencies and risks are budgeted. This budget can then be used as verification results become available, selecting individually the verification activity that provides the maximum expected value at each time-interval during the system development. This dynamic scheme for contracting verification activities is consistent with the mathematical properties of verification and guarantees efficiency and effectiveness of the verification strategy at completion of the system development.

The results of this research also suggest a need for methods that can cope with scalability issues. The application cases that have been used in this project are significantly simpler than those faced in actual acquisition programs, in terms of the size of the problem: number of parameters to be verified, number of potential verification activities, and number of time-intervals in which verification may be executed. The two approximation methods that have been explored have provided promising results. Future work is necessary to assess their performance at different levels of complexity.

Finally, simple approximations have been used to model rework activities and contractual structures. In terms of rework, two main limitations have been used for simplicity. Predefined rules for initiating rework activities have been used throughout the study. In realistic settings however, default protocols for rework activities may yield suboptimal actions. Future work is necessary to include formal decision points after verification results are available to determine if a rework activity is worth executing or not. Furthermore, the rework models employed in this project model the effect of rework as an action that brings the system back to its desired state before the verification activity was executed. However, such a model captures the effects of repair activities (i.e., solving a manufacturing problem), not those of rework activities. Rework activities change the internal structure of the system. Therefore, they impact the prior information relevant to its correct operation and make the model of the verification strategy for the system obsolete. Future work is necessary to incorporate accurate effects of rework in the models of verification strategies.

In terms of modeling contractual mechanism, this project has assumed that each verification can be simply paid for independently. However, this is in general not the case. For example, some verification activities require initial investment before they can be employed. Therefore, considering a verification activity at a future time-interval may require a partial expenditure at an earlier time-interval. This dependency has an impact on the way in which the set of verification strategies evolve with time. Future work is necessary to accurately model this type of verification development dependencies. Furthermore, the contractual structure that can guarantee fixed prices for a flexible contract over long development times is not evident. Premiums or incentives may be necessary to incorporate contract flexibility. Pricing and granularity of contractual options will also likely play a significant role in constraining the extent to which verification strategies can be adapted. Future work is necessary to consider these aspects.

Conclusions

This report presents a transformative approach to design verification strategies and contract verification activities. Instead of relying on a verification strategy that remains static during a system's development, the approach presented in this report suggests that the content of the verification strategy, and hence the contracting of its verification activities, need to be performed dynamically as the results of verification activities become available. This approach enables the acquisition of more affordable systems, while also improving the confidence on their correct operation, by facilitating the selection of necessary verification activities and the de-scoping of unnecessary ones.

The approach presented in this report leverages the concept of set-based design and applies it to the design of verification strategies. Furthermore, the approach builds upon a mathematical framework that describes the properties of verification. Because verification activities provide information about the system, the value of verification activities cannot be determined in absolute terms, but it is a function of the knowledge at the time of executing the verification activity. Therefore, past results play a major role in determining the value (and hence necessity) of future verification activities.

This new conceptualization departs from existing practice and the state of the art. The approach has been tested on several cases, representing the partial verification of a space instrument. The cases have confirmed the adequacy of the presented set-based design method to design and contract verification strategies. Overall, in all cases the application of the set-based design method yielded verification strategies with higher expected value than those yielded by using existing practice. Furthermore, the issue of scalability has also been explored in this project. Given the size of realistic systems and acquisition structures, the solution space of verification strategies cannot be exhaustively explored. Monte Carlo and Parallel Tempering have been studied as approximation methods to explore the solution space with promising results.

Finally, the research has already resulted in one published paper for the 2019 Acquisition Research Symposium, one published paper for the 2019 INCOSE International Symposium, and one published paper in Wiley's Systems Engineering journal. Two other journal papers resulting from this work are currently under preparation and will be submitted before the end of 2019.

Appendix – Data Used in Application Cases

Case 1. Proof-of-concept.

Table A1.1 $P(\theta_1)$

θ_1	Probability assignment
Error	0.8
No Error	0.2

Table A1.2 $P(\theta_2 | \theta_1)$

θ_1	θ_2	Probability assignment
Error	Error	0.8
Error	No Error	0.2
No Error	Error	0.1
No Error	No Error	0.9

Table A1.3 $P(V_1 | \theta_1)$

θ_1	V_1	Probability assignment
Error	Fail	0.7
Error	Pass	0.3
No Error	Fail	0.2
No Error	Pass	0.8

Table A1.4 $P(V_2 | V_1, \theta_1)$

θ_1	V_1	V_2	Probability assignment
Error	Fail	Fail	0.9
Error	Fail	Pass	0.1
Error	Pass	Fail	0.6
Error	Pass	Pass	0.4
No Error	Fail	Fail	0.2
No Error	Fail	Pass	0.8
No Error	Pass	Fail	0.1
No Error	Pass	Pass	0.9

Table A1.5 $P(V_3 | \theta_2)$

θ_2	V_3	Probability assignment
Error	Fail	0.9
Error	Pass	0.1
No Error	Fail	0.4
No Error	Pass	0.6

Table A1.6 $P(V_4 | V_3, \theta_2)$

θ_1	V_1	V_2	Probability assignment
Error	Fail	Fail	0.9

Error	Fail	Pass	0.1
Error	Pass	Fail	0.4
Error	Pass	Pass	0.6
No Error	Fail	Fail	0.3
No Error	Fail	Pass	0.7
No Error	Pass	Fail	0.1
No Error	Pass	Pass	0.9

Table A1.7. Cost Values

V_i	$C_V(V_i)$ ($k)	T_j	C_R at T_j ($k)	System error	$C_I(\theta_k)$ ($k)
V_1	50	1	100	θ_2	60,000
V_2	100	2	200		
V_3	75	3	300		
V_4	200	4	400		
		5	500		
		6	600		

Case 2. Improved model of rework.

Table A2.1. Conditional Probability Table for system parameter

θ_C	θ_S	$P(\theta_S \mid \theta_C)$
Error	Error	0.79
Error	No Error	0.21
No Error	Error	0.27
No Error	No Error	0.73

Table A2.2. Prior probabilities of the component parameter

θ_C	$P(\theta_C)$
Error	0.20
No Error	0.80

Table A2.3. Conditional Probability Table for verification activity V_1

θ_C	V_1	$P(V_1 \mid \theta_C)$
Error	Fail	0.73
Error	Pass	0.27
No Error	Fail	0.05
No Error	Pass	0.95

Table A2.4. Conditional Probability Table for verification activity V_2

θ_S	V_2	$P(V_2 \mid \theta_S)$
Error	Fail	0.85
Error	Pass	0.15
No Error	Fail	0.18
No Error	Pass	0.82

Case 3. Extending the size of the verification strategy: A Monte Carlo approach

Table A3.1. Partial dataset of the characterization of the verification network in Figure 8

Node No.	CPT							
1	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
2	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
3	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.94
4	0.03	0.97						
5	0.29	0.71						
6	0.95	0.95	0.87	0.92	0.92	0.87	0.83	0.73
7	0.18	0.82						
8	0.09	0.91						
9	0.24	0.76						
10	0.02	0.98						
11	0.27	0.73						
12	0.85	0.05	0.15	0.95				
13	0.95	0.05	0.05	0.95				
14	0.6	0.22	0.4	0.78				
15	0.81	0.07	0.19	0.93				
16	0.34	0.66						
17	0.27	0.73						
18	0.22	0.78						
19	0.02	0.98						
20	0.33	0.67						
21	0.08	0.92						
22	0.95	0.88	0.82	0.52	0.05	0.12	0.18	0.48
23	0.83	0.34	0.17	0.66				
24	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.92
25	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.92
26	0.83	0.05	0.17	0.95				
27	0.95	0.95	0.87	0.05	0.05	0.05	0.13	0.95
28	0.18	0.82						
29	0.08	0.92						
30	0.93	0.85	0.85	0.5	0.07	0.15	0.15	0.5
31	0.95	0.78	0.83	0.26	0.05	0.22	0.17	0.74
32	0.95	0.2	0.05	0.8				
33	0.95	0.95	0.95	0.74	0.95	0.95	0.95	0.7
34	0.36	0.64						
35	0.95	0.95	0.95	0.91	0.95	0.76	0.95	0.43
36	0.9	0.9	0.69	0.58	0.1	0.1	0.31	0.42
37	0.95	0.82	0.95	0.5	0.05	0.18	0.05	0.5
38	0.64	0.22	0.36	0.78				
39	0.81	0.23	0.19	0.77				
40	0.95	0.95	0.82	0.14	0.05	0.05	0.18	0.86
41	0.69	0.4	0.31	0.6				
42	0.94	0.82	0.79	0.32	0.06	0.18	0.21	0.68
43	0.95	0.79	0.89	0.29	0.05	0.21	0.11	0.71
44	0.95	0.9	0.95	0.32	0.05	0.1	0.05	0.68
45	0.91	0.09	0.09	0.91				
46	0.82	0.11	0.18	0.89				
47	0.8	0.11	0.2	0.89				
48	0.73	0.66	0.27	0.34				
49	0.65	0.43	0.35	0.57				
49	0.74	0.13	0.26	0.87				

Case 4. Extending the size of the verification strategy: A state-based approach

Table A4.1. Activity Cost

N 22	*N 23*	*N 24*	*N 25*	*N 26*	*N 27*	*N 28*	*N 29*	*N 30*	*N 31*
350	800	300	250	300	350	350	550	450	650
N 32	*N 33*	*N 34*	*N 35*	*N 36*	*N 37*	*N 38*	*N 39*	*N 40*	*N 41*
300	700	250	700	450	550	250	250	800	1000
N 42	*N 43*	*N 44*	*N 45*	*N 46*	*N 47*	*N 48*	*N 49*	*N 50*	
450	650	950	950	250	250	400	850	250	

Table A4.2. Basic Rework Cost

N 22	*N 23*	*N 24*	*N 25*	*N 26*	*N 27*	*N 28*	*N 29*	*N 30*	*N 31*
1800	1000	2200	1800	1800	1600	1000	2200	1400	1200
N 32	*N 33*	*N 34*	*N 35*	*N 36*	*N 37*	*N 38*	*N 39*	*N 40*	*N 41*
2600	2200	1000	2000	1000	2600	3200	2000	2600	1000
N 42	*N 43*	*N 44*	*N 45*	*N 46*	*N 47*	*N 48*	*N 49*	*N 50*	
2000	1000	1800	1200	1000	2400	1400	1000	1000	

THIS PAGE LEFT INTENTIONALLY BLANK

References

Bernstein, J. I. (1998). *Design methods in the aerospace industry: looking for evidence of set-based practices.* (MSc), Massachusetts Institute of Technolog, Boston, MA, USA.

Blanchard, B. S., & Fabrycky, W. J. (1990). *Systems engineering and analysis* (Vol. 4): Prentice Hall New Jersey;.

Earl, D. J., & Deem, M. W. (2005). Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics, 7*(23), 3910-3916. doi:10.1039/B509983H

Engel, A. (2010). *Verification, Validation, and Testing of Engineered Systems*. Hoboken, NJ: John Wiley & Sons, Inc.

Hansen, E., & Muller, G. (2012). 11.3.1 Set-based design – the lean tool that eludes us; Pitfalls in implementing set-based design in Kongsberg Automotive. *INCOSE International Symposium, 22*(1), 1603-1618. doi:doi:10.1002/j.2334-5837.2012.tb01425.x

Kennedy, B. M., Sobek, D. K., & Kennedy, M. N. (2014). Reducing Rework by Applying Set-Based Practices Early in the Systems Engineering Process. *Systems Engineering, 17*(3), 278-296. doi:doi:10.1002/sys.21269

Rapp, S., Chinnam, R., Doerry, N., Murat, A., & Witus, G. (2018). Product development resilience through set-based design. *Systems Engineering, 21*(5), 490-500. doi:doi:10.1002/sys.21449

Raudberget, D. (2010). Practical Applications of Set-Based Concurrent Engineering in Industry. *Journal of Mechanical Engineering, 56*(11), 685.

Salado, A. (2015). Defining Better Test Strategies with Tradespace Exploration Techniques and Pareto Fronts: Application in an Industrial Project. *Systems Engineering, 18*(6), 639-658. doi:10.1002/sys.21332

Salado, A., & Kannan, H. (2018a). A mathematical model of verification strategies. *Systems Engineering, 21*, 583-608.

Salado, A., & Kannan, H. (2018b). *Properties of the Utility of Verification*. Paper presented at the IEEE International Symposium in Systems Engineering, Rome, Italy.

Salado, A., & Kannan, H. (2019). Elemental Patterns of Verification Strategies. *Systems Engineering, In press*.

Salado, A., Kannan, H., & Farkhondehmaal, F. (2018). *Capturing the Information Dependencies of Verification Activities with Bayesian Networks*. Paper presented at the Conference on Systems Engineering Research (CSER), Charlottesville, VA, USA.

Singer, D. J., Doerry, N., & Buckley, M. E. (2009). What Is Set-Based Design? *Naval Engineers Journal, 121*(4), 31-43. doi:10.1111/j.1559-3584.2009.00226.x

Small, C., Buchanan, R., Pohl, E., Parnell, G. S., Cilli, M., Goerger, S., & Wade, Z. (2018). A UAV Case Study with Set-based Design. *INCOSE International Symposium, 28*(1), 1578-1591. doi:doi:10.1002/j.2334-5837.2018.00569.x

Wang, C., Hyman, J. D., Percus, A., & Caflisch, R. (2009). Parallel tempering for the traveling salesman problem. *International Journal of Modern Physics C, 20*(04), 539-556. doi:10.1142/s0129183109013893

Xu, P., & Salado, A. (2019). *A Concept for Set-based Design of Verification Strategies.* Paper presented at the INCOSE International Symposium, Orlando, FL, USA.

Acquisition Research Program
Graduate School of Business & Public Policy
Naval Postgraduate School
555 Dyer Road, Ingersoll Hall
Monterey, CA 93943

www.acquisitionresearch.net