2019-04-30

Towards the Dynamic Contracting of Verification Activities With Set-Based Design: An Initial Model of Rework

Xu, Peng; Salado, Alejandro

Monterey, California. Naval Postgraduate School

SYM-AM-19-050

PROCEEDINGS
OF THE
SIXTEENTH ANNUAL ACQUISITION RESEARCH SYMPOSIUM

WEDNESDAY SESSIONS
VOLUME I

Acquisition Research:
Creating Synergy for Informed Change

May 8–9, 2019

Published: April 30, 2019

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

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

Towards the Dynamic Contracting of Verification Activities With Set-Based Design: An Initial Model of Rework

Peng Xu—is a PhD student with the Grado Department of Industrial and Systems Engineering at Virginia Tech. He received his MS in mechanical engineering from National Cheng Kung University in 2015 and his BS in mechanical engineering from Shandong University in 2013. His research interests include complex system diagnosis, dynamic decision making, and knowledge elicitation. [xupeng@vt.edu]

Alejandro Salado—is an Assistant Professor with the Grado Department of Industrial and Systems Engineering at Virginia Tech. His research focuses on applying decision analysis to improve the practice of engineering, in particular in the areas of verification and validation, and on improving problem formulation through modeling. Dr. Salado is a recipient of the NSF CAREER Award and the Fulbright International Science and Technology Award. He holds a BSc and an MSc in electrical engineering (Polytechnic University of Valencia), an MSc in project management and an MSc in electronics engineering (Polytechnic University of Catalonia), the SpaceTech MEng in space systems engineering (Delft University of Technology), and a PhD in systems engineering (Stevens Institute of Technology). [asalado@vt.edu]

Abstract

This paper is intended to disseminate initial outcomes of the NPS Research Acquisition Program "Dynamic Contracting of Verification Activities by Applying Set-Based Design to the Definition of Verification Strategies" project. Verification activities provide the evidence of contractual fulfillment. 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. This research project shows that contractually committing to a fixed verification strategy at the beginning of an acquisition program fundamentally leads to suboptimal acquisition performance. This is caused by the uncertain nature of system development, which will make, as it progresses, 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. Such an approach to contracting may be enabled by applying set-based design to the definition of verification strategies. This paper provides a summary of such an approach and contributes with a refined model of rework activities that may be undertaken to increase the confidence on the proper functioning of the system as verification results become known.

Introduction

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 as

> 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 the 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 (CRs), 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 a 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).

This paper presents the current state of the research project and contributes with a refined model of rework activities that may be undertaken to increase the confidence on the proper functioning of the system as verification results become known.

Background: Models of Verification Strategies

Primary Characteristics of 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),

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

where S_A is a set of system characteristics, $B_t(S_A, t_n)$ 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 $B_t(S_A, t_n)$" (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 paper, 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). As explained in the introduction of this paper, 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 are 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 paper, 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 $\pi(\theta \mid \mathbf{s})$ based on his/her prior belief distribution $\pi(\theta)$ and the density function $f(\mathbf{v} \mid \theta)$, 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 $D_\theta = \{(a,b) : a,b \in \theta_Z, f(b \mid \mathbf{a}) \neq f(b)\}$.

  $B = (\{\theta_Z, V\}, D_\Upsilon)$ is a simple directed graph that captures the ability of the verification activities to provide information about one or more system parameters, where $D_\Upsilon = \{(a,b) : a \in \theta_Z, b \in V, f(b \mid \mathbf{a}) \neq f(b)\}$.

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 paper) 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 paper; it has been used here for consistency with the original source.

A Concept for Dynamic Contracting of Verification Activities

The concept for dynamic contracting of verification activities has been presented in Xu and Salado (2019) and is depicted in Figure 1 in comparison with the current approach. The following description is reproduced verbatim from the original source:

> 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.
>
> 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. (Xu & Salado, 2019)

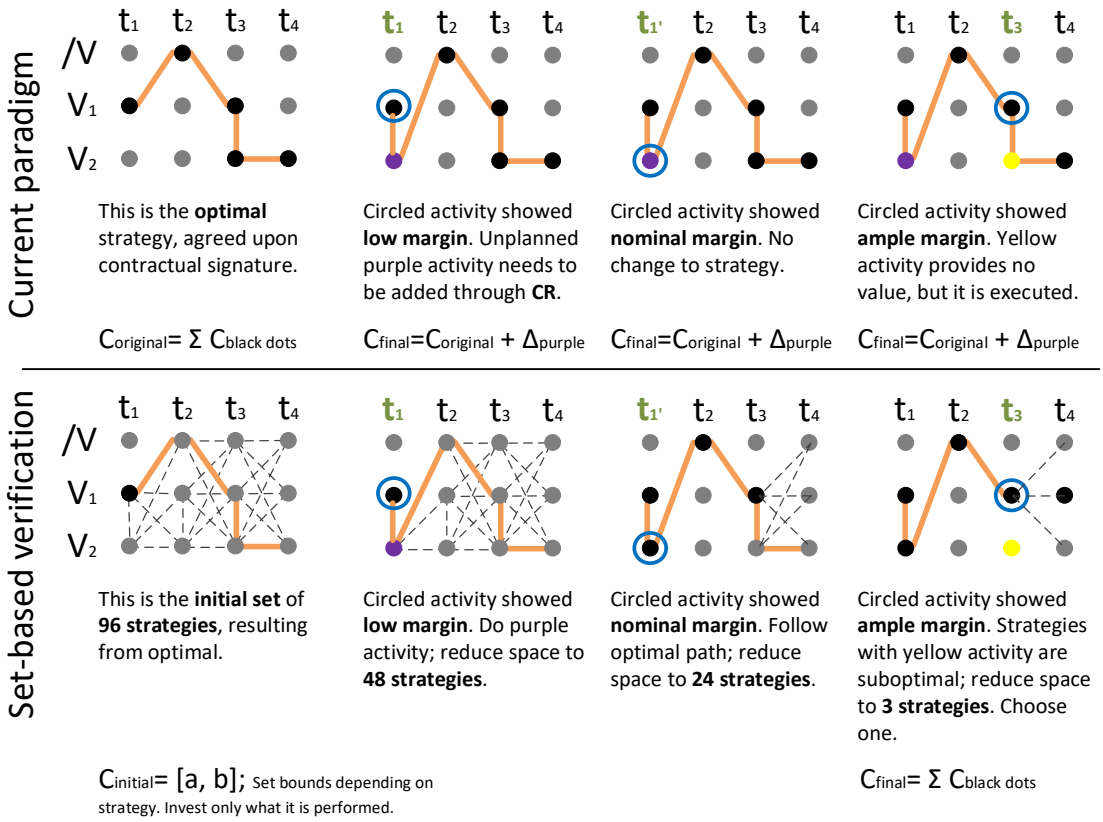

Figure 1. Current vs. Set-Based Approaches for Designing Verification Strategies

(Xu & Salado, 2019)

Note. C: cost of executing verification; t_i: verification events; /V: no verification; V_i: verification activity.

Applying Set-Based Design to the Design of Verification Strategies

The 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 makes product development resilient against changes in external factors is given in Rapp et al. (2018). The approach has been successfully applied in the conceptual stages of naval systems (Singer, Doerry, & Buckley, 2009), graphic industry products (Raudberget, 2010), automotive products (Raudberget, 2010), and aeronautic systems (Bernstein, 1998), among others.

As discussed in the introduction, these findings informed the application of set-based design to the design of verification strategies (Xu & Salado, 2019). The benefits of set-based design were explored in a notional case study with synthetic data. Results indicated that set-based approach yielded higher expected value. In addition, set-based design seemed to respond faster to adjusting its parameters than the benchmark when receiving information from verification evidence, which indicates "the benchmark approach is inefficient when compared against the proposed set-based approach" (Xu & Salado, 2019). Further research is necessary to confirm these findings, though.

The basic process proposed 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 2. 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.

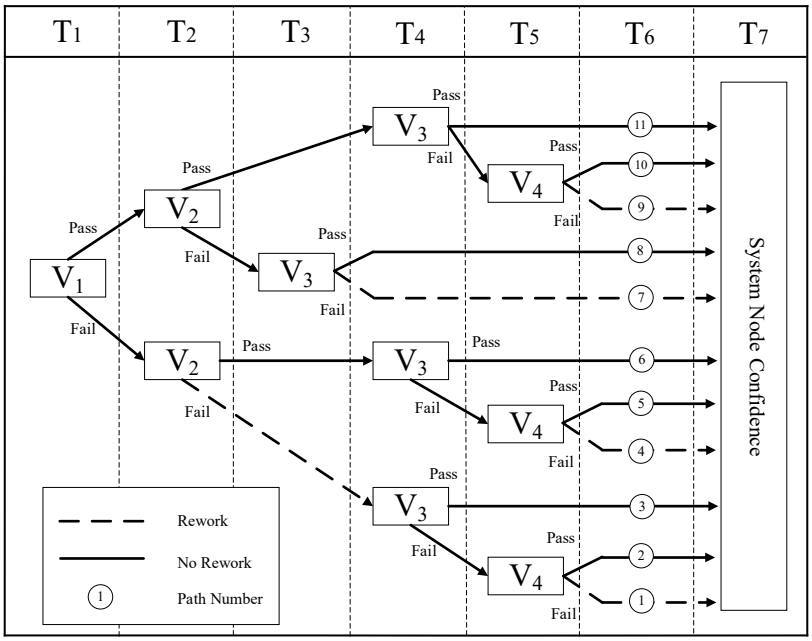

Figure 2. Verification Path Tree
(Xu & Salado, 2019)

Overall in this example, 11 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.

A Refined Model of Rework

Background

In prior work, rework has been treated as a predefined decision based on the achieved confidence (Xu & Salado, 2019). Specifically, if the confidence in the correct functioning of the system (for example, as represented by parameter θ in the section entitled Mathematical Models of Verification Strategies) would fall below a certain threshold, then a rework activity was considered to be executed automatically. In this paper, we present a model of rework activities that considers a different decision mechanism. In particular, a rework activity is initiated if a verification activity fails.

Problem Statement

Consider the simple overarching verification network in Figure 3. 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).

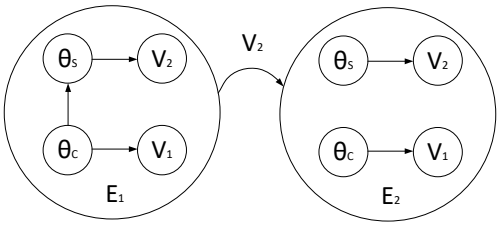

Figure 3. Overarching Verification Network

Five verification strategies can be devised by leveraging the overarching network (notation from Salado and 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 is assumed that S_5 is not meaningful, and therefore it will not be further considered.

The cost to execute a verification activity is denoted by σ_V. Table 1 lists the cost to execute each verification strategy. It is assumed that no overlap exists in the cost of executing the verification activities.

Table 1. 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 2 lists the expected costs of impact for each strategy. It is assumed that $\sigma_I = 10,000\text{K}$.

Table 2. Impact Cost of Deploying the System With an Error

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

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

Model of Rework Cost

Rework cost is denoted by σ_R. The key aspect is that the cost of rework will depend 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 is assumed that rework results in a state of knowledge equivalent to $V = p$. This is because in the theoretical framework used in this paper, system attributes are not accessible; the only verification evidence is Salado and Kannan (2019).

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

- For S_1, $E[\sigma_R(S_1)] = 0$ because, since there is no verification activity executed, errors cannot be found and rework activities initiated.

- For S_2, $E\left[\sigma_R\left(S_2\right)\right] = P\left(V_1 = \neg p\right) \cdot \sigma_R\left(C,C\right)$, where $\sigma_R\left(A,B\right)$ indicates that rework happens for assembly *A* when integrated at assembly level *B*. In this case, $\left(C,C\right)$ 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 $\sigma_R\left(C,C\right)$ 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 is 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 the system level. However, based on experience, it has been assumed that de-integration activities are less preferred. Under these conditions, the expected rework cost for S_3 is given by Equation 2:

$$E\left[\sigma_R\left(S_3\right)\right] = P\left(V_2 = f\right) \cdot \left[\sigma_R\left(S,S\right) + P\left(\theta_S = e, \theta_C = e \,|\, V_2 = f\right) \cdot \sigma_R\left(C,S\right)\right]. \qquad (2)$$

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 is 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 at the 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 Equation 3:

$$E\left[\sigma_R(S_4)\right] = P(V_1 = f)\cdot\left(\sigma_R(C,C) + P(V_2 = f \mid V_1 = p)\cdot\sigma_R(S,S)\right) + P(V_1 = p)\cdot P(V_2 = f \mid V_1 = p)\cdot \left(\sigma_R(S,S) + P(\theta_S = e, \theta_C = e \mid V_2 = f, V_1 = p)\cdot\sigma_R(C,S)\right) \quad (3)$$

Table 3 lists the corresponding rework cost used in the model.

Table 3. Rework Costs

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

Input Data

Cost figures are synthetic and given in the previous section, Model of Rework Cost. Probability assignments use synthetic data and are given in Tables 4 through 7. Following the modeling approach presented in Salado and Kannan (2019), prior beliefs are 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 are created for the verification activity nodes. Posterior beliefs are 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.

Table 4. 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 5. Prior Probabilities of the Component Parameter

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

Table 6. 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 7. 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

Results and Discussion

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 is only used to explore the application of the refined rework model, so the case is still useful.

Results are shown in Figure 4. 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 set up 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 is rework at higher integration levels more expensive, but also, there is a chance that the problem at system level is caused by a problem at component level. Such de-integration effort considerably increases the resulting rework cost.

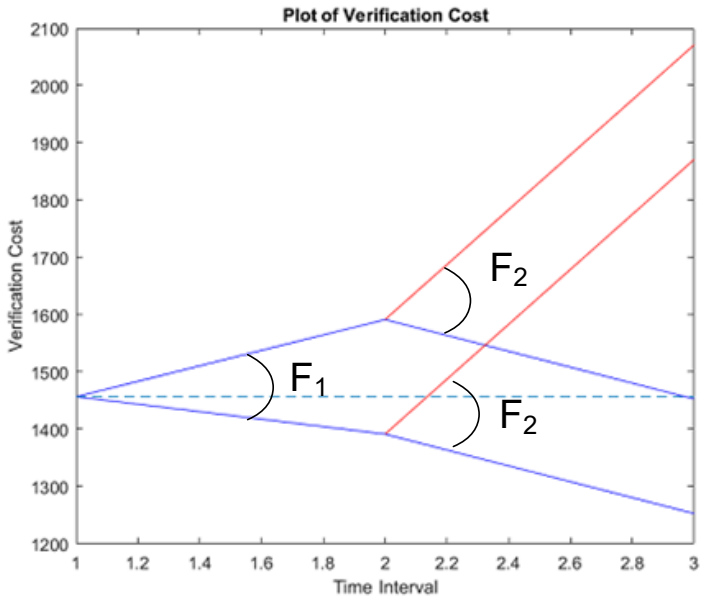

Figure 4. Plot of Verification Paths

Conclusions

This paper has shown that current approaches to contracting verification strategies in acquisition programs conflict with the inherent nature of verification. As a result, verification strategies in acquisition programs are set to suboptimality. This paper supports the idea of using dynamic contracting to overcome this problem. In this proposed approach, contracting of verification activities is spread throughout the system development. Instead of pre-agreeing on a fixed set of activities, verification activities are contracted at different points during the development. In this way, the results of prior verification activities can be used to determine the optimal path going forward in the system development.

Set-based design, which has been successfully applied in conceptual design and system architecture, provides a conceptual framework that can enable the dynamic contracting of verification strategies. Exploratory prior work seems to indicate that the set-based approach is stronger than the current paradigms for contracting of verification to deal with the uncertain nature of system development, yielding strategies of higher expected value. This paper has synthesized the process to apply set-based design to verification strategies and pointed to how evaluating dominance of strategies may be helpful to deal with the complexity resulting from the size of the problem.

Finally, this paper has also presented a refined model of the effects of rework activities in the expected value of a verification strategy. Although the model is still not sufficiently accurate of a real-life scenario, it improves prior work. Specifically, prior work relied on a predefined rework decision based on confidence thresholds. Instead, the proposed model considers that a rework activity is always initiated when a verification activity fails and considers its effect a function of the likelihood of such a verification result. In addition, and more importantly, it also incorporates the notion that rework may be needed at different levels of integration, requiring different levels of investment to solve the problem.

It should be noted that the effort is ongoing and is planned to be completed within the timeframe of the NPS Acquisition Research Program's "Dynamic Contracting of

Verification Activities by Applying Set-Based Design to the Definition of Verification Strategies" project.

References

Bernstein, J. I. (1998). *Design methods in the aerospace industry: Looking for evidence of set-based practices.* Cambridge, MA: Massachusetts Institute of Technology.

Engel, A. (2010). *Verification, validation, and testing of engineered systems*. Hoboken, NJ: John Wiley & Sons.

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: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.

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

Xu, P., & Salado, A. (2019). *A concept for set-based design of verification strategies.* Paper presented at the INCOSE International Symposium, Orlando, FL.

Acknowledgements & Disclaimer

This material is based upon work supported by the Acquisition Research Program under HQ00341810002. 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 Department of Defense, nor does mention of trade names, commercial practices, or organizations imply endorsement by the U.S. Government.

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL
555 DYER ROAD, INGERSOLL HALL
MONTEREY, CA 93943

www.acquisitionresearch.net