Proceedings of the ASME 2020
International Design Engineering Technical Conferences
and Computers and Information in Engineering Conference
IDETC/CIE 2020
August 16-19, 2020, St. Louis, MO, USA

IDETC2020-22582

IS VERIFYING FREQUENTLY AN OPTIMAL STRATEGY? A BELIEF-BASED MODEL OF VERIFICATION

Aditya U. Kulkarni[1], Alejandro Salado[1], Christian Wernz[2], Peng Xu[1]

[1]Virginia Tech, Blacksburg, VA, USA
[2]Virginia Commonwealth University, Richmond, VA, USA

ABSTRACT

Verification activities increase an engineering team's confidence in its system design meeting system requirements, which in turn are derived from stakeholder needs. Conventional wisdom suggests that the system design should be verified frequently to minimize the cost of rework as the system design matures. However, this strategy is based more on experience of engineers than on a theoretical foundation. In this paper, we develop a belief-based model of verification of system design, using a single system requirement as an abstraction, to determine the conditions under which it is cost effective for an organization to verify frequently. We study the model for a broad set of growth rates in verification setup and rework costs. Our results show that verifying a system design frequently is not always an optimal verification strategy. Instead, it is only an optimal strategy when the costs of reworking a faulty design increase at a certain rate as the design matures.

Keywords: Belief-based modeling, system verification

NOMENCLATURE

T	number of development phases
t	generic development phase
s_t	state of the system design
c_t	verification setup cost
r_t	expected cost correcting design errors
r_F	expected cost of project failure
β_t	belief value in ideal state of design
ε	confidence retention factor
d_t	decision to verify or not verify
b_t	belief vector
P_v	belief transformation matrix for verify
P_{-v}	belief transformation matrix for not-verify
$l_{t,v}$	cost vector associated with verification
$l_{t,-v}$	cost vector associated with not verifying
$d^*_t(\beta_t)$	optimal decision given belief value
$V_t(b_t, d_t)$	optimal cost function in development phase t

1. INTRODUCTION

Verification activities seek to determine if a system design meets the system requirements. These system requirements define the space of acceptable system solutions (or system designs) [1], which the engineering teams explore during the design process to arrive at the final system design. Through verification activities, engineering teams can check if the current system design is in the space of acceptable system solutions [2], that is, verification activities are the means by which engineers check if the system is being built "right" [3]. Furthermore, verification activities enable design teams to detect errors early in the design development, where an error is a violation of one or more system requirements. Early error detection prevents unwanted rework costs for the design teams in the future [4]. Thus, verification activities have been recognized as an integral part of the system design process [5].

Verifying the system design frequently has been previously advocated both in industry and research literature [6-8]. However, this strategy is derived more from experience than from a theoretical foundation. Since verification activities are cost and time intensive, verifying the system design more frequently than necessary can result in misallocation of limited resources [9]. An optimal verification strategy is one that balances the cost of verification activities with the risks associated with erroneous designs [10]. In this regard, a significant challenge is to theoretically test whether it is optimal to verify a system design frequently throughout the design process.

In this paper, we lay the foundation to theoretically determine the conditions under which frequent verification is an optimal strategy in systems design. Our work explores if frequent verification of a single system requirement by an organization is an optimal strategy, or not, for different scenarios. Each scenario is defined by growth rates in the setup costs to execute verification activities and expected costs of corrective actions taken to address problems in the system design potentially discovered through the execution of verification activities. We develop a belief-based model of verification in which the organization's confidence in the system design meeting its system requirements is modeled using belief distributions. The organization's optimal verification strategy is then defined as the one that minimizes the organization's expected verification costs as a function of its belief in the true state of its design.

2. BACKGROUND

Prior works on verification in systems engineering can be broadly classified into conceptual approaches [11-14], empirical approaches [4, 15-18], and probabilistic models of verification [19, 20]. We define as conceptual approaches to verification those that are based on guidelines, industry standards and best practices. These guidelines are developed by studying the failures of past projects. Problems are identified and then correlated with verification activities that could have been performed to identify early the observed failures. Though conceptual approaches to verification activities have their merits, a significant drawback is that they are rooted more in experience and hindsight than mathematical analysis. Without a proper understanding of the scientific foundation of verification, adopting a conceptual approach to verification can result in suboptimal verification strategies [21].

Empirical works on verification incorporate more mathematical formalism by considering the costs and risks associated with verification activities [4]. In this approach, pre-defined verification strategies are applied in different systems engineering projects, and the results of the same are observed. The goal is then to use results from a wide range of systems engineering projects and empirically deduce the characteristics of optimal verification strategies. The drawback of empirical approaches is that their solutions are only as good as the cases they explore [22]. This prevents the results of empirical studies from being extended to those systems engineering projects that lie outside the original dataset.

The drawbacks of conceptual and empirical approaches to verification are significantly reduced when probabilistic models of verification are used [23, 24]. In this approach, the verification process is assumed to reveal the errors in design according to a predefined stochastic process. Examples include the software reliability growth model to detect errors in software design [25, 26], the canonical model of verification for systems engineering projects [9, 10, 27], and models for sequential and parallel testing in new product development [28-30]. Though probabilistic models of verification are grounded in mathematical analysis, their use of stochastic processes in modeling verification activities restricts them to those scenarios where there is prior data by which the parameters of a model's stochastic process can be determined.

It has been previously recognized that designing systems is a cognitive process [31]. Recent works [32] have identified that engineers make design decisions based on subjective beliefs about the true state of the system design. When engineers are uncertain about whether their current design meets system requirements, it is more due to a lack of confidence in how all their design activities have combined than due to a belief in the design process being inherently stochastic [33]. Thus, traditional probability frameworks are not suited for capturing the epistemic uncertainty in the system design process, and belief-based frameworks are more appropriate [34]. Yet, the current body of literature on verification in systems engineering largely ignores the epistemic aspect of systems design.

To the best of our knowledge, only the recent works by Salado et al. [35-38] have explicitly captured the epistemic uncertainty in the design process by using beliefs to model verification strategies. In their work, verification strategies are derived based on the organization's changing belief in the system design meeting the system requirements. Using a belief-based approach to derive verification strategies is advantageous since beliefs better represent an organization's knowledge in the current state of its design, and this facilitates a more accurate representation of the risk vs reward tradeoff in determining optimal verification strategies. We leverage this conceptualization of verification strategies in this paper.

3. MODEL

3.1 Model environment

The system design process is often considered to progress from a conceptual design, to preliminary design to detailed design [39]. We model the system design process as a series of design development phases, or simply development phases. We say that two development phases are distinct if the system designs at the end of the two phases are different in either design attributes or design maturity. Here, design maturity refers to the state of the current design, with blueprints and modeling representing designs of low maturity and operational prototypes representing designs of high maturity.

We divide each development phase into two periods: design period and verification period. In the design period, design activities are executed to take the current system design to the next level of maturity, whereas in the verification period, verification activities are carried out to determine whether the current design meets the system requirements. Examples of design activities include analysis, modeling, tradespace studies, mock-ups, prototyping, and manufacturing. Examples of verification activities include testing, inspection, demonstration, and analysis. We assume that design activities will be executed in each development phase, whereas verification activities will be executed based on a strategy (that is, they may be executed in some development phases and not in others).

Since design activities are assumed to be executed in each development phase, we normalize all costs associated with design activities to \$0. For verification activities, we restrict our attention to two high-level verification costs in each development phase: setup cost for the verification activity in a development phase and the expected cost to correct a faulty design in a development phase, which we refer to as rework cost. In addition to these two high-level verification costs, we assume the organization incurs an expected cost of project failure if the system design does not meet system requirements at the end of all development phases.

In reality, there may be multiple organizations working on a system design that has numerous requirements, with many of these requirements being correlated [40]. To develop a model of verification that is analytically tractable, in this paper, we restrict our attention to a single organization and a single system requirement, which we refer to as the requirement of interest. For modeling simplicity, we assume that the organization classifies the state of the design in each development phase as either the system design satisfying the requirement of interest, or not satisfying it.

To quantify the organization's confidence in the system design meeting the requirement of interest in each development phase, we use belief distributions [41-43]. Belief distributions enable us to model the organization's subjective assessment of its design meeting the requirement of interest. Per Dempster-Shafer theory of evidence, belief distributions are transformed by the evidence received from the environment [43]. In our model, we implicitly account for evidence provided by the environment, in this case the system design process, by defining the transformation of the organization's beliefs when it executes design and verification activities. Specifically, we assume that the organization's belief in the system design meeting the requirement of interest *decreases* when design activities are executed, due to changes in the design attributes. Conversely, we assume that the organization's belief in the system design meeting the requirement of interest *increases* through verification activities either by obtaining successful verification results or by performing a corrective action (i.e., rework) when errors are found.

The organization's goal is then to utilize its beliefs in the current state of the system design to determine an optimal verification strategy. The optimal verification strategy is defined as one that minimizes the verification setup costs, expected costs of correcting a faulty design upon verification, and the expected cost of failure for the organization.

3.2 Model parameters

Let T denote the number of development phases. To denote a generic development phase, we use the subscript $t \in \{1, \ldots, T\}$. As mentioned before, we assume the organization broadly classifies the state of the system design as either meeting the requirement of interest or not meeting the requirement of interest. To denote the state of the system design *at the start* of development phase *t*, we use the state variable $s_t \in \{0,1\}$, where $s_t = 1$ denotes the system design meeting the requirement of interest, and $s_t = 0$ denoting that the system design does not meet the requirement of interest. We will refer to $s_t = 1$ as the ideal state, and $s_t = 0$ as the non-ideal state of the system design.

In development phase *t,* we denote the organization's decision by $d_t \in \{v, -v\}$, where v denotes the organization verifies the system design, and $-v$ denotes the organization does not verify the system design. The setup cost for verification activities is denoted by c_t and the expected cost to correct any errors in system design upon verification, or the rework cost, is denoted by r_t. We assume that if the organization chooses to verify its design in development phase *t,* then it will incur the setup cost c_t for certain, whereas, the rework cost r_t is only incurred if the system design is found to be in the non-ideal state upon verification. Furthermore, we denote the expected cost of project failure by r_F. To avoid exploring trivial scenarios, in this paper, we assume $c_t + r_t < r_F$ for all *t*. Hence, it is in the organization's interest to verify the system design in at least one development phase.

We model the organization's knowledge in the state of its design using belief distributions. The organization's belief in the system design being in the *ideal state* at the start of development phase *t,* or $s_t = 1$, is denoted by β_t. In our model, we assume the belief values over the state space add up to 1. This implies the organization's belief value in system design being in the *non-ideal state* at start of development phase *t* is equal to $1 - \beta_t$. Furthermore, belief values adding up to 1 in our model implies $\beta_t \in [0,1]$.

As per our model assumption, if the organization does not verify the design in the current development phase, then the changes in design with respect to the previous development phase will decrease the organization's belief in the system design being in the ideal state at the end of the current development phase. To model this, we assume that β_t is transformed by a factor ε to $\beta_t \varepsilon$ after all the design activities are carried out in development phase *t.* We refer to ε as the confidence retention factor. The parameter ε is a measure of the organization's confidence in the correctness of its design activities (in other words, ε can be understood as the predictability of the organization's design process). In this paper, we assume that $0 < \varepsilon < 1$. This implies that our model assumes the organization will not discover any error in the design during the design process because we consider design activities to not generate new information about the state of the system [44]. Hence, its belief in the ideal state of the system design always reduces after design activities.

The value of the confidence retention factor ε is a measure of the rate of refinement in the system design with each passing development phase. A low value of ε indicates that the system design undergoes major refinement from conceptual design to detailed design, whereas a high value of ε indicates the system design undergoes minor refinement from conceptual design to detailed design. In this paper, we restrict our attention to those scenarios where the confidence retention factor ε is stationary in all development phases. However, in practice, it is possible for the value of ε to be dependent on either the verification strategy or the design maturity.

For mathematical tractability, we restrict our analysis to those scenarios where verification activities reveal the true state of the system design and result in the organization having complete confidence in the ideal state of the system design at that developmental stage. In general, this is not true, and verification activities may only provide partial confidence in the system design meeting the requirement of interest.

To model our assumption of verification activities leading to complete confidence in the ideal state of the system design, we say that if the organization chooses to verify its design in development phase *t,* then $\beta_{t+1}=1$, i.e., the organization has complete confidence in the ideal state of the system design at the start of development phase $t+1$. Similarly, if the organization chooses not to verify the design in development phase *t,* then its belief in the ideal state of the system design at the start of development phase $t+1$ is $\beta_{t+1}=\beta_t\varepsilon$. The evolution of the organization's belief in development phase *t* is graphically represented in Figure 1.

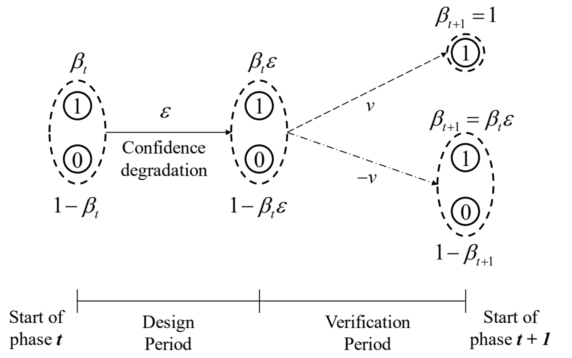

FIGURE 1: EVOLUTION OF THE ORGANIZATION'S BELIEFS IN DEVELOPMENT PHASE ***t***

For ease of discussion, we will utilize vector notation for the remainder of this paper. To this end, let the row vector $b_t=(1-\beta_t,\beta_t)$ denote the vector of beliefs for the organization at the start of development phase *t,* where the first element of vector b_t denotes the organization's belief in the non-ideal state of the system design, and the second element of vector denotes the organization's belief in the ideal state of the system design. To represent the transformation of b_t into b_{t+1} for the two alternatives of the verification decision $d_t\in\{v,-v\}$, we define the following matrices: $\mathbb{P}_{-v}=\begin{pmatrix}1 & 0\\ 1-\varepsilon & \varepsilon\end{pmatrix}$ and $\mathbb{P}_{v}=\begin{pmatrix}0 & 1\\ 0 & 1\end{pmatrix}$.

Then,

$$b_{t+1}=\begin{cases}b_t\mathbb{P}_{-v} & \text{if } d_t=-v\\ b_t\mathbb{P}_{v} & \text{if } d_t=v\end{cases}. \tag{1}$$

Let the column vectors $l_{t,v}$ and $l_{t,-v}$ denote the organization's vector of costs associated with its decisions to verify and not verify, respectively. The organization's cost vector for verifying the system design in any development phase *t* is defined by $l_{t,v}=(c_t+r_t,c_t)'$, where the apostrophe denotes a vector transpose. Based on our model assumptions, the organization incurs no cost for not verifying the system design in development phases $t<T$. Thus, for all $t<T$, $l_{t,-v}=(0,0)$. At the end of the final development phase, if the system design is in the non-ideal state, then the organization incurs the expected cost of failure. We capture this by defining the organization's cost vector for not verifying the system design in the final development phase *T* as $l_{T,-v}=(-r_F,0)'$.

3.3 Optimal verification strategy based on beliefs

The organization can follow a pre-defined verification plan $(d_1,\ldots,d_T)$, which defines a decision $d_t\in\{v,-v\}$ in each development phase. However, a pre-defined verification plan ignores the organization's changing belief in the true state of its design, and can end up dictating the organization verify its design even when the organization's confidence in the ideal state of the design is high [45]. In order to minimize verification costs over the entire development, the organization must implement a verification strategy, or a dynamic verification plan that specifies a decision in each development phase while accounting for the organization's belief in the true state of its design in each development phase [45].

Before we present a formal definition of an optimal verification strategy, we elucidate two important features of a pre-defined verification plan $(d_1,\ldots,d_T)$. First, given the initial belief vector b_1, a verification plan completely defines the organization's belief vectors in all development phases. This is so, since the belief vector b_2 is the result of the transformation of the initial belief vector b_1 through the decision d_1. The belief vector b_3 is the result of the transformation of b_2 through the decision d_2, and so on until the end of all development phases. Second, the decision d_t in a verification plan determines the organization's immediate costs in development phase *t*, and also affects the organization's verification costs in phases $t+1,\ldots T$. This follows from the belief vector b_{t+1} being a transformation of b_t through d_t, b_{t+2} being a transformation of b_{t+1} through d_{t+1}, and so on.

Using the two properties of the verification plan mentioned above, we now define the organization's optimal decision in development phase *t*, denoted by $d_t^*(b_1)\in\{v,-v\}$, as the one that minimizes the organization's expected verification costs over the development phases $t,t+1,\ldots,T$ given the organization's initial belief vector is b_1. Our definition is motivated by the fact that if the organization's decision in development phase *t* affects the overall costs over the development phases $t,t+1,\ldots,T$, then the rational decision for the organization is the one that minimizes the expected costs over the development phases $t,t+1,\ldots,T$. Using the definitions and properties mentioned above, we now define the optimal verification strategy for the organization as the set of optimal decisions $\{d_1^*(b_1),\ldots,d_T^*(b_1)\}$ in each development phase.

To determine $d_t^*(b_1)$, the organization must know its immediate and future expected costs resulting from a decision d_t and the belief transformation of b_t into b_{t+1} through the decision d_t. To this end, we define $V_t(b_t, d_t)$ as the lowest possible cost the organization can expect to incur over development phases $t, t+1, \ldots, T$ for the decision d_t and belief vector b_t. We will refer to $V_t(\cdot)$ as the organization's *optimal cost function* in development phase *t*.

The optimal cost functions $V_t(\cdot)$ can be determined through dynamic programming [46]. In the last development phase *T,* the organization only needs to consider the immediate costs associated with the decision d_T, and hence the organization's optimal cost functions are defined by

$$V_T(b_T, v) = b_T \mathbb{P}_{-v}\ l_{T,v}\text{, and} \tag{2}$$

$$V_T(b_T, -v) = b_T \mathbb{P}_{-v}\ l_{T,-v}\,. \tag{3}$$

In development phase *T-1,* the organization's optimal cost function $V_{T-1}(\cdot)$ must consider the effects of the organization's decision d_{T-1} on the expected costs in development phase *T.* Since, Equations (2) and (3) define the minimum cost the organization can expect to incur in development phase *T* for any belief vector b_T for each possible decision $d_T \in \{v, -v\}$, then to determine $V_{T-1}(\cdot)$, the organization only needs to determine the immediate expected cost of decision d_{T-1}, determine the transformation of b_{T-1} into b_T as a result of the decision d_{T-1}, and then use $V_T(b_T, v)$ and $V_T(b_T, -v)$ to determine the minimum cost it can expect to incur in development phase *T.* Hence, in development phase *T-1,* the organization's optimal cost function for its decision to verify is

$$V_{T-1}(b_{T-1}, v) = b_{T-1}\mathbb{P}_{-v}\ l_{T-1,v} + \min\{V_T(b_{T-1}\mathbb{P}_v, v), V_T(b_{T-1}\mathbb{P}_v, -v)\} \tag{4}$$

and for its decision to not verify it is

$$V_{T-1}(b_{T-1}, -v) = \min\{V_T(b_{T-1}\mathbb{P}_{-v}, v), V_T(b_{T-1}\mathbb{P}_{-v}, -v)\}\,. \tag{5}$$

The organization's optimal cost functions for all other development phases can be determined in the same manner as the one presented above for development phase *T-1.* For $t < T$, assume that the organization has already determined the optimal cost functions $V_{t+1}(b_{t+1}, v)$ and $V_{t+1}(b_{t+1}, -v)$ for development phase $t+1$. Then, in development phase *t,* the organization's optimal cost function for its decision to verify is defined by

$$V_t(b_t, v) = b_t \mathbb{P}_{-v}\ l_{t,v} + \min\{V_{t+1}(b_t\mathbb{P}_v, v), V_{t+1}(b_t\mathbb{P}_v, -v)\} \tag{6}$$

and for its decision to not verify is defined by

$$V_t(b_t, -v) = \min\{V_{t+1}(b_t\mathbb{P}_{-v}, v), V_{t+1}(b_t\mathbb{P}_{-v}, -v)\}\,. \tag{7}$$

With the knowledge of the optimal cost functions for all development phases, the organization can then determine the set of optimal decision functions $\{d_1^*(b_1), \ldots, d_T^*(b_1)\}$. We illustrate the procedure to determine $d_t^*(\cdot)$ for a possible initial belief vector $\hat{b}_1$. In phase 1, the optimal decision for the organization is defined by

$$d_1^*(\hat{b}_1) = \arg\max_{d_1 \in \{v,-v\}} \{V_1(\hat{b}_1, d_1)\}\,. \tag{8}$$

If $d_1^*(\hat{b}_1) = v$, then we know $\hat{b}_2 = \hat{b}_1\mathbb{P}_v$, else $\hat{b}_2 = \hat{b}_1\mathbb{P}_{-v}$ holds true. Since we know the organization's starting belief in phase 2 is $\hat{b}_2$, it follows that

$$d_2^*(\hat{b}_1) = \arg\max_{d_2 \in \{v,-v\}} \{V_2(\hat{b}_2, d_2)\}\,. \tag{9}$$

We then determine $\hat{b}_3$ from $d_2^*(b_1)$ and $\hat{b}_2$, and the process repeats for all remaining development phases.

The formulation presented above to determine the organization's optimal verification strategy $\{d_1^*(b_1), \ldots, d_T^*(b_1)\}$ is similar in structure to partially observable Markov decision processes (POMDPs) [47]. However, unlike POMDPs, our model implicitly accounts for the observations. For a given set of parameter values for our model, the optimal verification strategy for the organization can be numerically determined using standard solution algorithms for POMDPs after adjusting for the differences in our model and the structure of POMDPs.

It is analytically intractable to derive a closed form expression for $d_t^*(b_1)$ that does not involve the maximum function. Instead, we will seek to solve our model numerically for a broad class of setup and rework cost functions to derive insights on the optimal verification strategy. To so, we will use the solution algorithm presented in the Appendix.

4. IS FREQUENT VERIFICATION OPTIMAL?

Frequent verification with respect to our model means that the organization verifies the system design in all development phases, irrespective of its belief in the true state of its system design. In this section, we derive the conditions under which frequent verification is an optimal strategy for the organization. We begin with a numerical illustration of our model for six different sets of values for the model parameters c_t, r_t and r_F. The purpose of the numerical illustration is to show that frequent verification is not an optimal strategy in general. After the numerical illustration, we will derive theoretical lower bounds for r_t and r_F that must be satisfied to ensure frequent verification is an optimal strategy for the organization.

4.1 Numerical illustration: when frequent verification *is not* optimal

The change in c_t and r_t with respect to t can be broadly defined as: 1) constant, 2) linearly increasing, and 3) exponentially increasing. Even if c_t is constant for the requirement of interest, in most real-world cases, r_t will increase with t as the design matures. Hence, we study the remaining six combinations of the types of increase in c_t and r_t. We will refer to each combination of the type of increase in c_t and r_t as a case. The six cases are presented in Table 1.

TABLE 1: DESCRIPTION OF CASES FOR NUMERICAL ILLUSTRATION

Case	*Type of increase in* c_t	*Type of increase in* r_t
1	Constant	Linear
2	Constant	Exponential
3	Linear	Linear
4	Linear	Exponential
5	Exponential	Linear
6	Exponential	Exponential

For the numerical illustration we notionally consider $T = 6$ and $\varepsilon = 0.7$. As mentioned before, we assume that the expected cost of failure $r_F > c_t + r_t$. Since r_t increases with *t* in all cases, we select the value of r_F to ensure $r_F > c_T + r_T$. The notional values of c_t, r_t and r_F for each case are listed in Table 2.

TABLE 2: NOTIONAL PARAMETER VALUES

Case	c_t	r_t	r_F
1	\$5000	\$6000 + \$1000**t*	\$100,000
2	\$5000	\$7000* t^2	\$300,000
3	\$4000 + \$1000**t*	\$6000 + \$1000**t*	\$300,000
4	\$4000 + \$1000**t*	\$7000* t^2	\$300,000
5	\$5000* t^2	\$6000 + \$1000**t*	\$1,000,000
6	\$5000* t^2	\$7000* t^2	\$1,000,000

The optimal verification strategies for the six cases were numerically determined using the solution algorithm presented in Section 3.4 with the aid of MATLAB®. For all six cases, Figure 2 graphs the optimal decision function for the organization, $d_t^*(\cdot)$, for each of the 6 development phases (*on the y-axis in each graph*) given the organization's initial belief in the ideal state of the system design, β_1 (*on the x-axis for each graph*).

As shown in Figure 2, for cases 1, 3, and 5, the optimal verification strategy for the organization is to verify the system design in the final development phase, but not in any other development phase, irrespective of the organization's initial belief in the ideal state of the system design, β_1. In case 2, the optimal verification strategy requires the organization to verify in the last development phase, in the first three development phases depending on the value of β_1, but not in development phases four and five. In case 4, the optimal verification strategy for the organization is to verify the system design only in development phases one, three and six. Finally, in case 6, the optimal verification strategy for the organization is to verify in first and last development phase, but not in any other development phase.

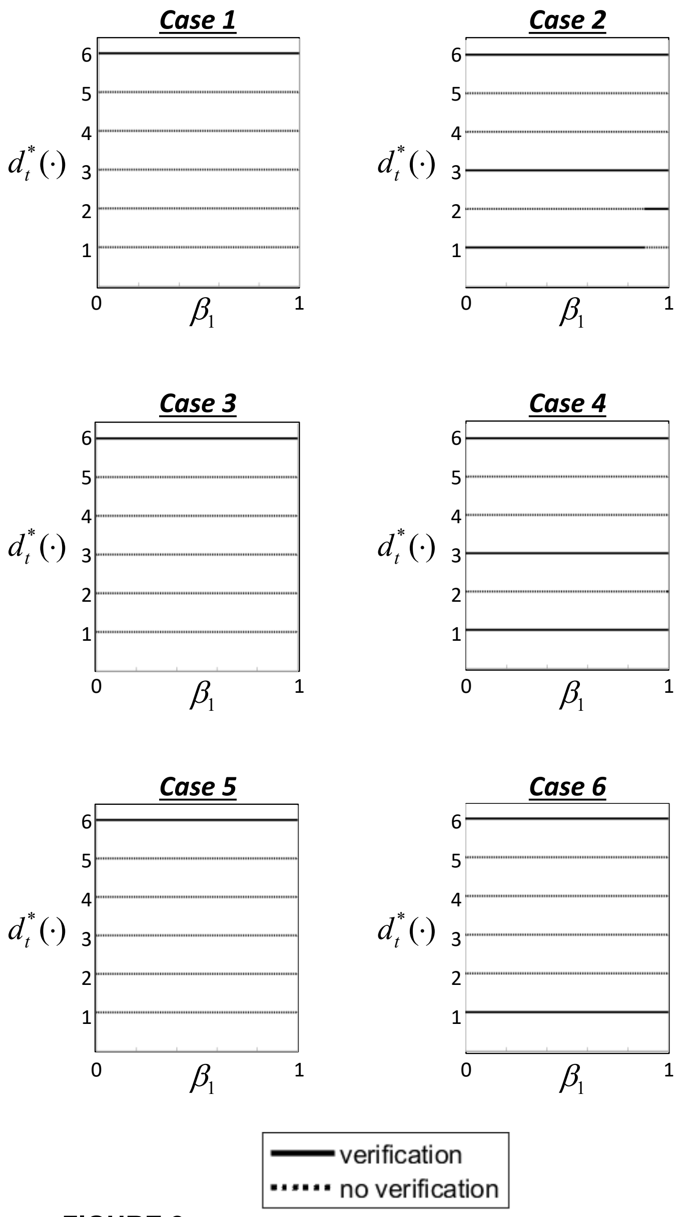

FIGURE 2: OPTIMAL VERIFICATION STRATEGY FOR 6 NOTIONAL CASES

The results of our numerical example suggest that when the expected cost to correct the errors in design, r_t, increases linearly, as in cases 1, 3 and 5, the optimal verification strategy is to verify the system design once, in the final development phase. Only when r_t increases exponentially with each development phase, as in cases 2, 4 and 6, does our model suggest that verifying the system design in a development phase other than the final one can potentially be part of the optimal verification strategy.

We hypothesize the cause of our numerical results with the following examples. For the results of cases 1, 3 and 5, consider an optical instrument with a requirement of interest being the cleanliness of its observing surface. The setup cost for verifying whether the sensor meets the requirement of interest may be constant, linearly increasing or exponentially increasing, depending on the design maturity. Due to the nature of the requirement of interest, the cost of correction during integration through to deployment just involves cleaning the surface. In this scenario, our model suggests that the requirement of interest for the sensor is to be verified once, in the final development phase, to maximize the organization's final confidence in the instrument meeting the requirement of interest and minimizing verification costs.

For cases 2, 4 and 6, consider a satellite's solar panel with the requirement of interest being that the panel provides certain level of electrical power. The electrical power that the solar panel will provide will depend on several factors, such as surface area of the panel, efficiency of the solar cells, and survivability ratio after vibration testing. Hence, choosing incorrect surface dimensions for the solar panel in early phases may lead to costly rework later in later development phases. Thus, verifying whether the panel surface will accommodate sufficient solar cells to meet the electrical power provision requirement should probably be carried out in early development phases. However, if the organization is confident of its design process to determine panel surface so as to not requiring costly panel resizing in later development phases, then perhaps it is not cost-effective for the organization to verify the solar panel surface (i.e., requirement of interest) in every development phase.

4.2 When *is* frequent verification optimal?

The results of the numerical example presented in Section 4.1 leads to the following question: if continuous verification of the system design is not a requirement enforced by stakeholders, *under what conditions is it optimal for the organization to verify the system design in all development phases irrespective of its belief in the system design meeting the requirement of interest*? To answer this question, we now study our model analytically.

Let $b_{t,0}$ and $b_{t,1}$ denote the first and second element of belief vector b_t, respectively. Consider development phase *T*. The organization's optimal cost functions for its two decisions in development phase *T* are

$$V_T(b_{T,1}=\beta_T, v) = -c_T - r_T(1-\beta_T\varepsilon) \text{, and} \tag{10}$$

$$V_T(b_{T,1}=\beta_T, -v) = -r_F(1-\beta_T) \,. \tag{11}$$

The organization will verify the system design in development phase *T* only if $V_T(b_{T,1}=\beta_T, v) \ge V_T(b_{T,1}=\beta_T, -v)$

$$\Rightarrow -c_T - r_T(1-\beta_T\varepsilon) \ge -r_F(1-\beta_T)$$

$$\Rightarrow \beta_T \le \frac{1}{\varepsilon}\Big(1-\frac{c_T}{r_F-r_T}\Big) = \beta_T^* \,. \tag{12}$$

That is, the optimal verification strategy for the organization in the final development phase is to verify the system design if $\beta_T \le \beta_T^*$. If $\beta_T^* \ge 1$, then it is optimal for the organization to verify the system design in the final development phase for all values of β_T. In turn, $\beta_T^* \ge 1$ only if $\frac{1}{\varepsilon}\Big(1-\frac{c_T}{r_F-r_T}\Big) \ge 1$

$$\Rightarrow r_F \ge r_T + c_T + \frac{c_T\varepsilon}{1-\varepsilon} \,. \tag{13}$$

Now, assume that Condition (13) is satisfied, and it is optimal for the organization to verify the system design in the final development phase. Consider development phase *T-1*. With the prior knowledge that the organization will verify the system design in the final development phase, the organization's optimal cost functions associated with its two decisions in development phase *T-1* are

$$V_{T-1}(b_{T-1,1}=\beta_{T-1}, v) = -c_{T-1} - r_{T-1}(1-\beta_{T-1}\varepsilon) - c_T - r_T(1-\varepsilon) \,, \tag{14}$$

and

$$V_{T-1}(b_{T-1,1}=\beta_{T-1}, -v) = -c_T - r_T(1-\beta_{T-1}\varepsilon^2) \,. \tag{15}$$

The organization will verify the system design only if
$V_{T-1}(b_{T-1,1}=\beta_{T-1}, v) \ge V_{T-1}(b_{T-1,1}=\beta_{T-1}, -v)$

$$\Rightarrow -c_{T-1} - r_{T-1}(1-\beta_{T-1}\varepsilon) - c_T - r_T(1-\varepsilon) \ge -c_T - r_T(1-\beta_{T-1}\varepsilon^2)$$

$$\Rightarrow \beta_{T-1} \le \frac{1}{\varepsilon}\Big(1-\frac{c_{T-1}}{r_T\varepsilon - r_{T-1}}\Big) = \beta_{T-1}^* \,. \tag{16}$$

Once again, the verifying the system design in development phase *T-1* is always an optimal strategy only if $\beta_{T-1}^* \ge 1$. This in turn implies $\frac{1}{\varepsilon}\Big(1-\frac{c_{T-1}}{r_T\varepsilon - r_{T-1}}\Big) \ge 1$

$$\Rightarrow r_T \ge \frac{1}{\varepsilon}\Big(r_{T-1} + c_{T-1} + \frac{c_{T-1}\varepsilon}{1-\varepsilon}\Big) \,. \tag{17}$$

We see that it is optimal for the organization to verify the system design in development phase *T-1* only if the expected repair cost in development phase *T* is greater than the lower bound defined by Condition (17). Else, $\beta_{T-1}^* \le 1$ and it will be optimal for the organization to not verify the design for high values of β_{T-1}.

Using the procedure presented above for development phase *T-1,* we can derive the condition under which it is optimal for the organization to verify the system design in development phase $t < T$. To this end, assume that it is optimal for the organization to verify the system design in development phase $t+1$. Then, the optimal cost functions in development phase *t* are

$$V_t(b_{t,1} = \beta_t, v) = -c_t - r_t(1-\beta_t\varepsilon) - c_{t+1} - r_{t+1}(1-\varepsilon)\text{ , and} \tag{18}$$

$$V_t(b_{t,1} = \beta_t, -v) = -c_{t+1} - r_{t+1}(1-\beta_t\varepsilon^2)\,. \tag{19}$$

The organization will verify the system design if
$V_t(b_{t,1} = \beta_t, v) \ge V_t(b_{t,1} = \beta_t, -v)$

$$\Rightarrow \beta_t \le \frac{1}{\varepsilon}\Big(1 - \frac{c_t}{r_{t+1}\varepsilon - r_t}\Big) = \beta_t^*\,. \tag{20}$$

Hence, verifying in development phase *t* is always an optimal strategy if $\beta_t^* \ge 1$

$$\Rightarrow r_{t+1} \ge \frac{1}{\varepsilon}\Big(r_t + c_t + \frac{c_t\varepsilon}{1-\varepsilon}\Big)\,. \tag{21}$$

From Conditions (17) and (21), we see that the lower bound on the expected repair cost is recursive in nature. Using induction [48], we can further reduce Condition (21) as

$$r_{t+1} \ge \frac{1}{\varepsilon}\Big(r_t + c_t + \frac{c_t\varepsilon}{1-\varepsilon}\Big)$$

$$\Rightarrow r_{t+1} \ge \frac{1}{\varepsilon}\Big(\frac{1}{\varepsilon}\Big(r_{t-1} + c_{t-1} + \frac{c_{t-1}\varepsilon}{1-\varepsilon}\Big) + c_t + \frac{c_t\varepsilon}{1-\varepsilon}\Big)$$

$$r_{t+1} \ge \frac{r_1}{\varepsilon^t} + \frac{1}{(1-\varepsilon)}\sum_{j=1}^{t}\frac{c_j}{\varepsilon^{t+1-j}}\,. \tag{22}$$

Conditions (13) and (22) impose a theoretical lower bound on the values of r_F and r_t for frequent verification of the system design to be an optimal verification strategy for the organization. Previously, we had assumed that the expected cost of failure r_F is greater than $c_t + r_t$ for all *t*. However, Condition (13) places a stronger restriction for verification in the final development phase to be an optimal strategy for all belief values β_T: it is necessary for r_F to be greater than $c_T + r_T$ by a margin that is at least equal to $c_T\varepsilon/(1-\varepsilon)$. Similarly, Condition (22) suggests that for frequent verification to be optimal, r_t must be far greater than the sum of the expected repair cost in development phase 1 and the setup costs in development phases $1,\ldots,t-1$.

Consider again the 6 cases explored for the numerical illustration in section 4.1. We see that the notional values for r_F satisfy Condition (13) for all six cases. Whereas, the notional values for r_t do not satisfy Condition (22) for all cases. Hence, we conclude that verification in the final development phase being optimal for all cases in the numerical illustration resulted from Condition (13) being satisfied.

Though Condition (22) was not satisfied for all cases, our numerical results did not suggest that verification of the system design in early development phases was sub-optimal. Indeed, in cases 2, 4 and 6, where r_t increased exponentially, our results suggest that verification in development phases other than the last one is part of the optimal verification strategy. *This leads us to conjecture that as r_t approaches the theoretical bound defined by Condition (22), verification in early development phases becomes optimal* for different belief values, and when r_t satisfies Condition (22), verification in all development phases of the system design is optimal.

It is reasonable to assume that in most real-world scenarios, the expected cost of failure, r_F, will satisfy Condition (13), and it is optimal for the organization to verify the system design in the final development phase. However, it is not necessary for Condition (22) to be satisfied in all development phases. In such scenarios, it is perhaps better to adopt a belief-based approach to verification of the system design than follow conventional wisdom and risk misallocating limited resources to "over-verifying" the system design.

5. CONCLUSION

In this paper, we used a belief-based model to determine if verifying a system frequently, using a single requirement as an abstraction, is an optimal strategy. The results of our analysis suggest that verifying the system design frequently is not an optimal strategy in general, and it is optimal only when the expected cost to correct errors in the system design increase at a certain rate as the design matures. Two important limitations of our model are that we assume that the organization's confidence in the correctness of its design activities (or predictability of its design process) does not change as the design matures and we focus only on a single system requirement. At this point, we conjecture that relaxing the aforementioned constraints might expand the space of scenarios where frequent verification is optimal.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. CMMI-1762883 and CMMI-1762336.

APPENDIX

Assume that c_t, r_t, r_F and ε are given. Let $\phi_{T,v} = \mathbb{P}_{-v}l_{T,v}$ and let $\phi_{T,-v} = \mathbb{P}_v l_{T,-v}$. Then, in phase *T,* the organization's optimal cost functions can be defined as

$$V_T(b_T, v) = \min(b_T\phi_{T,v})\text{ , and} \tag{23}$$

$$V_T(b_T, -v) = \min(b_T\phi_{T,-v})\,. \tag{24}$$

The minimum function returns the lowest value of its vector argument and its use is redundant in equations (23) and (24), but this formulation will prove useful for the remaining development phases.

In development phase $T-1$, the organization's optimal cost function for its decision to verify can be defined as

$$V_{T-1}(b_{T-1},v) = b_{T-1}\mathbb{P}_{-v}\ l_{T-1,v} + \min\{\, V_T(b_{T-1}\mathbb{P}_v, v), V_T(b_{T-1}\mathbb{P}_v, -v)\}$$

$$\Rightarrow V_{T-1}(b_{T-1},v)$$

$$= \min\{b_{T-1}\cdot(\mathbb{P}_{-v}l_{T-1,v} + \mathbb{P}_v\phi_{T,v}), b_{T-1}\cdot(\mathbb{P}_{-v}l_{T-1,v} + \mathbb{P}_v\phi_{T,-v})\}\,.$$

Define $\phi_{T-1,v} = [(\mathbb{P}_{-v}l_{T-1,v} + \mathbb{P}_v\phi_{T,v}),(\mathbb{P}_{-v}l_{T-1,v} + \mathbb{P}_v\phi_{T,-v})]$. Then, optimal cost function for verifying the design in development phase $T-1$ can be redefined as

$$V_{T-1}(b_{T-1},v) = \min(b_{T-1}\phi_{T-1,v})\,. \tag{25}$$

Similarly, the optimal cost to not verify the system design in development phase $T-1$ can be redefined as

$$V_{T-1}(b_{T-1},-v) = \min(b_{T-1}\phi_{T-1,-v})\,, \tag{26}$$

where $\phi_{T-1,-v} = [\mathbb{P}_{-v}\phi_{T,-v}, \mathbb{P}_{-v}\phi_{T,v}]$.

Proceeding in manner presented above for development phase $T-1$, assume that the matrices $\phi_{t+1,v}$ and $\phi_{t+1,-v}$ have been previously computed. Then, the organization's optimal cost functions in development phase *t* are defined by

$$V_t(b_t,v) = \min(b_t\phi_{t,v})\text{, and} \tag{27}$$

$$V_t(b_t,-v) = \min(b_t\phi_{t,-v})\,, \tag{28}$$

where $\phi_{t,v} = [(\mathbb{P}_{-v}l_{t,v} + \mathbb{P}_v\phi_{t+1,v}),(\mathbb{P}_{-v}l_{t,v} + \mathbb{P}_v\phi_{t+1,-v})]$ and $\phi_{t,-v} = [\mathbb{P}_{-v}\phi_{t+1,-v}, \mathbb{P}_{-v}\phi_{t+1,v}]$.

By computing the matrices $\phi_{t,v}$ and $\phi_{t,-v}$ for all development phases, we can determine the organization's optimal cost function value for any belief vector b_t, and this in turn enables us to determine the optimal verification strategy $\{d_1^*(b_1),\ldots,d_T^*(b_1)\}$. The space of all feasible initial belief vectors b_1 is uncountable. Hence, $\{d_1^*(b_1),\ldots,d_T^*(b_1)\}$ must be computed for a finite set of belief vectors $\Psi = \{b_1^1,\ldots,b_1^M\}$ that reasonably discretizes the space of initial belief vectors. The following table outlines the solution algorithm to numerically solve our model assuming all parameter values and the set Ψ is known beforehand.

TABLE 3: SOLUTION ALGORITHM

Initialize c_t, r_t, r_F, ε, $\mathbb{P}_v$, $\mathbb{P}_{-v}$, $l_{t,v}$, l_{t-v} and Ψ			
1	Set $\phi_{T,v} = \mathbb{P}_{-v}l_{T,v}$ and $\phi_{T,-v} = \mathbb{P}_{-v}l_{T,-v}$		
2	*For* $t = T-1$; $t > 0$		
	2a	Set $\phi_{t,v} = [(\mathbb{P}_{-v}l_{t,v} + \mathbb{P}_v\phi_{t+1,v}),(\mathbb{P}_{-v}l_{t,v} + \mathbb{P}_v\phi_{t+1,-v})]$	
	2b	Set $\phi_{t,-v} = [\mathbb{P}_{-v}\phi_{t+1,-v}, \mathbb{P}_{-v}\phi_{t+1,v}]$	
	2c	Set $t = t-1$, return to step 2	
3	*For each* $b_1^x \in \Psi$ *, Do*		
	3a	Set $\hat{b}_1 = b_1^1$	
	3b	*For* $t = 1$; $t \le T$	
		3b-i	Set $d_t^*(\hat{b}_1) = \arg\max_{d_t\in\{v,-v\}}\{\,V_t(\hat{b}_t, d_t)\}$
		3b-ii	If $t < T$, then set $\hat{b}_{t+1} = \begin{cases}\hat{b}_t\mathbb{P}_v & \text{if } d_n^*(\hat{b}_1) = v\\ \hat{b}_t\mathbb{P}_{-v} & \text{otherwise}\end{cases}$
		3b-iii	Set $t = t+1$, return to step 3b
	Return to step 3		

REFERENCES

1. Salado, A., R. Nilchiani, and D. Verma, *A contribution to the scientific foundations of systems engineering: Solution spaces and requirements.* Journal of Systems Science and Systems Engineering, 2017. **26**(5): p. 549-589.
2. Salado, A. and R. Nilchiani, *On the Evolution of Solution Spaces Triggered by Emerging Technologies.* Procedia Computer Science, 2015. **44**: p. 155-163.
3. INCOSE, *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. version 4.0 ed. 2015, Hoboken, NJ, USA: John Wiley and Sons, Inc.
4. Shabi, J., Y. Reich, and R. Diamant, *Planning the verification, validation, and testing process: a case study demonstrating a decision support model.* Journal of Engineering Design, 2017. **28**(3): p. 171-204.
5. Sage, A.P. and W.B. Rouse, *Handbook of systems engineering and management*. 2014: John Wiley & Sons.
6. Chang, T.-f., et al. *"Continuous verification" in mission critical software development*. in *Proceedings of the Thirtieth Hawaii International Conference on System Sciences*. 1997. IEEE.
7. Klingstam, P. and B.-G. Olsson. *Using simulation techniques for continuous process verification in industrial system development*. in *2000 Winter Simulation Conference Proceedings (Cat. No. 00CH37165)*. 2000. IEEE.
8. Maropoulos, P.G. and D. Ceglarek, *Design verification and validation in product lifecycle.* CIRP annals, 2010. **59**(2): p. 740-759.
9. Engel, A., *Verification, validation, and testing of engineered systems*. Vol. 73. 2010: John Wiley & Sons.

10. Engel, A. and M. Barad, *A methodology for modeling VVT risks and costs.* Systems Engineering, 2003. **6**(3): p. 135-151.

11. McGarry, F. and G. Page, *Performance evaluation of an independent software verification and integration process.* NASA Goddard, Greenbelt, MD, SEL Sill 0, 1982.

12. Nagano, S., *Space systems verification program and management process: Importance of Implementing a Distributed-Verification Program with Standardized Modular-Management Process.* Systems Engineering, 2008. **11**(1): p. 27-38.

13. Powell, P.B., *Software validation, verification, and testing technique and tool reference guide.* 1982.

14. Wallace, D.R. and R.U. Fujii, *Software verification and validation: an overview.* Ieee Software, 1989. **6**(3): p. 10-17.

15. Cook, T.D. and C.S. Reichardt, *Qualitative and quantitative methods in evaluation.* 1979.

16. Lee, A.S., *A scientific methodology for MIS case studies.* MIS quarterly, 1989: p. 33-50.

17. McCutcheon, D.M. and J.R. Meredith, *Conducting case study research in operations management.* Journal of Operations Management, 1993. **11**(3): p. 239-256.

18. Tahera, K., C.F. Earl, and C.M. Eckert, *Integrating virtual and physical testing to accelerate the engineering product development process.* IJITM, 2014. **13**(2/3): p. 154-175.

19. Barad, M. and A. Engel, *Optimizing VVT strategies: a decomposition approach.* Journal of the Operational Research Society, 2006. **57**(8): p. 965-974.

20. Yamada, S., T. Ichimori, and M. Nishiwaki, *Optimal allocation policies for testing-resource based on a software reliability growth model.* Mathematical and Computer Modelling, 1995. **22**(10-12): p. 295-301.

21. Schipper, S., *Diagnosing verification and validation problems in public civil engineering projects: How" building the right system right" can go wrong*. 2016, University of Twente.

22. Mendling, J., *Empirical studies in process model verification*, in *Transactions on petri nets and other models of concurrency II*. 2009, Springer. p. 208-224.

23. Ahmadi, R. and R.H. Wang, *Managing development risk in product design processes.* Operations Research, 1999. **47**(2): p. 235-246.

24. Goel, A.L. and K. Okumoto, *Time-dependent error-detection rate model for software reliability and other performance measures.* IEEE Transactions on Reliability, 1979. **28**(3): p. 206-211.

25. Hossain, S.A. and R.C. Dahiya, *Estimating the parameters of a non-homogeneous Poisson-process model for software reliability.* IEEE Transactions on Reliability, 1993. **42**(4): p. 604-612.

26. Ohba, M., *Inflection S-shaped software reliability growth model*, in *Stochastic Models in Reliability Theory*. 1984, Springer. p. 144-162.

27. Engel, A. and M. Last, *Modeling software testing costs and risks using fuzzy logic paradigm.* Journal of Systems and Software, 2007. **80**(6): p. 817-835.

28. Ha, A.Y. and E.L. Porteus, *Optimal timing of reviews in concurrent design for manufacturability.* Management Science, 1995. **41**(9): p. 1431-1447.

29. Loch, C.H., C. Terwiesch, and S. Thomke, *Parallel and sequential testing of design alternatives.* Management Science, 2001. **47**(5): p. 663-678.

30. Thomke, S. and D.E. Bell, *Sequential testing in product development.* Management Science, 2001. **47**(2): p. 308-323.

31. Ullman, D.G. and T.A. Dietterich, *Mechanical design methodology: implications on future developments of computer-aided design and knowledge-based systems.* Engineering with computers, 1987. **2**(1): p. 21-29.

32. Agarwal, H., et al., *Uncertainty quantification using evidence theory in multidisciplinary design optimization.* Reliability Engineering & System Safety, 2004. **85**(1-3): p. 281-294.

33. Thunnissen, D.P. *Uncertainty classification for the design and development of complex systems*. in *3rd annual predictive methods conference*. 2003. Newport Beach CA.

34. Christensen, B.T. and L.J. Ball, *Fluctuating epistemic uncertainty in a design team as a metacognitive driver for creative cognitive processes.* CoDesign, 2018. **14**(2): p. 133-152.

35. Salado, A. and H. Kannan, *A mathematical model of verification strategies.* Systems Engineering, 2018. **21**: p. 583-608.

36. Salado, A. and H. Kannan, *Elemental patterns of verification strategies.* Systems Engineering, 2019. **22**(5): p. 370-388.

37. Salado, A., H. Kannan, and F. Farkhondehmaal, *Capturing the Information Dependencies of Verification Activities with Bayesian Networks*, in *Conference on Systems Engineering Research (CSER)*. 2018: Charlottesville, VA, USA.

38. Xu, P. and A. Salado. *A Concept for Set-based Design of Verification Strategies*. in *INCOSE International Symposium*. 2019. Orlando, FL, USA.

39. Raymer, D., *Aircraft design: a conceptual approach*. 2012: American Institute of Aeronautics and Astronautics, Inc.

40. Sommerville, I. and P. Sawyer, *Requirements engineering: a good practice guide*. 1997: John Wiley & Sons, Inc.

41. Smets, P., *What is Dempster-Shafer's model.* Advances in the Dempster-Shafer theory of evidence, 1994: p. 5-34.

42. Smets, P., *The transferable belief model and other interpretations of Dempster-Shafer's model.* arXiv preprint arXiv:1304.1120, 2013.

43. Sentz, K. and S. Ferson, *Combination of evidence in Dempster-Shafer theory*. Vol. 4015. 2002: Citeseer.

44. Salado, A. *An Elemental Decomposition of Systems Engineering*. in *2018 IEEE International Systems Engineering Symposium (ISSE)*. 2018. IEEE.

45. Xu, P. and A. Salado. *A Concept for Set-based Design of Verification Strategies*. in *INCOSE International Symposium*. 2019. Wiley Online Library.

46. Bertsekas, D.P., et al., *Dynamic programming and optimal control*. Vol. 1. 1995: Athena scientific Belmont, MA.

47. Sondik, E.J., *The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs.* Operations Research, 1978. **26**(2): p. 282-304.

48. Graham, R.L., et al., *Concrete mathematics: a foundation for computer science.* Computers in Physics, 1989. **3**(5): p. 106-107.