Pliability and Viable Systems: Maintaining Value Under Changing Conditions

Pliability and Viable Systems: Maintaining Value under Changing Conditions

Brian Mekdeci, Adam M. Ross, Donna H. Rhodes, and Daniel E. Hastings,

***Abstract*— As systems become more complex, and have longer lifespans, they will likely encounter contextual variation or be themselves subject to change. Systems need to not only be feasible, but viable as well. That is, they need to be able to continue to provide value in spite of any potential exogenous or endogenous changes. Viability has been defined for other domains, but it has not been defined for engineered systems. This paper defines what it means for an engineered system to be viable, and shows that it is related to, but different from other existing "-ilities" such as survivability and reliability. This paper also addresses the need to ensure that endogenous changes do not inadvertently cause unintended interactions that harm the system overall. A new "-ility", pliability, is introduced that specifies the limits on how a system can change, without "breaking" or violating an architecture that was intended and validated. Like changeability, pliability increases robustness by allowing systems to voluntarily change in response to dynamic contexts, and increases survivability by increasing the likelihood that unintentional changes are still within the set of allowable architecture-defined instances. It also distinguishes allowable changes from those that would require additional validation, reducing the effort required to get those changes approved by a diverse set of stakeholders.**

***Index Terms*— pliability, survivability, system architecting, system engineering, systems of systems, reliability, resilience.**

I. Introduction

Unlike natural systems, engineered systems are designed to deliver value (i.e. the net benefit after costs are considered) to its stakeholders. The value that an engineered system delivers to its stakeholders depends on what the system is, what the system does, and contextual factors (e.g. the physical environment that surrounds the system, available resources, or stakeholder expectations). If nothing changes, then an engineered system that is providing acceptable value to its stakeholders will continue to do so. However, many engineered systems are expected to operate for long lifetimes within dynamic and challenging contexts, and numerous changes, both exogenous and endogenous to the system, can be expected. This is particularly true for system architects who must design Autonomous Vehicles (AV). AVs are particularly challenging to design because they are not only expected to do complex tasks in difficult environments, but also perform missions that often evolve over their lifetime. For example, AVs may be built and acquired for solo surveillance missions, but later be used as a communication relay in a larger System of Systems (SoS). As a result, it is not enough for AV system architects to design a "feasible" system, i.e. an engineered system that can deliver value to its stakeholders for a specific context. Rather, AV system architects face the non-trivial task of designing "viable" systems, which can deliver value to their stakeholders over long lifetimes that may include perturbations (defined as imposed state changes in the system's components, operations, or context which could jeopardize value delivery [1]). We can say that an engineered system is "viable" if it is expected to provide acceptable value to its stakeholders, over its life era, and "not viable" otherwise. We define *viability* for engineered systems as *the current likelihood that an engineered system will provide acceptable value to its stakeholders, over its life era*. The life era is both the expected time the system needs to last, as well as a sequence of epochs (pairs of contexts and needs) that it is expected to encounter (known as the "system era"). While viability may seem synonymous with other similar non-functional qualities like survivability, this paper will highlight the unique properties of a viability and suggest ways in which system architects can design viable systems by managing both exogenous and endogenous change.

A. Viability in Technical and Non-Technical Literature

Although not commonly used in the technical literature, the term "viability" is defined and used in numerous other disciplines. Viability for organizations is defined in the management cybernetics literature as "the survival or preservation of identity in a changing environment" [2]. This definition implies that an organization cannot change too much, in order for it to be viable, otherwise it simply becomes a new organization and the old organization would be considered "dead." In medicine, the term "viability" is typically applied when estimating the likelihood that a fetus will be able to survive outside its mother's womb. The key consideration in this case is the fetus' ability to survive the contextual change, and exist for a lifetime in a significantly more hostile environment. In the systems literature, which

This paper was submitted on August 08, 2013 for review and revised on November 25, 2013.

Brian Mekdeci was with the Massachusetts Institute of Technology as a doctoral student and research associate when this research was conducted, but is now a research fellow at the Defense and Systems Institute, University of South Australia in Adelaide SA, 5000 Australia (e-mail: brian.mekdeci@unisa.edu.au).

Daniel E. Hastings (email: hastings@mit.edu), Adam M. Ross (email: ross@mit.edu) and Donna H. Rhodes (email: rhodes@mit.edu) are all currently with the Massachusetts Institute of Technology, Cambridge, MA 02139 USA.

includes natural and artificial systems, Beer [3] developed the Viable System Model (VSM), to describe how a system can continue to exist and maintain its identity in a dynamic environment. The basis behind the VSM is that a viable system must satisfy Ashby's Law of Requisite Variety [4], by maintaining adequate control mechanisms to cope with uncertainty in the environment.

There is no exact agreement on what viability means in these disciplines, but a few important constructs emerge. All of the definitions include "survival" or the idea of a continued existence, over some period of time, in spite of contextual changes. Systems may need to change, as the definition from systems literature suggests, in order for them to deal with contextual change, but they should not have to change too much, as the organizational literature suggests, otherwise the original system would be considered not viable. In the next section, the technical literature is examined for change-related "-ilities" to help define and differentiate viability for engineered systems.

II. Change-Related "-ilities"

There has been a considerable amount of research done at determining qualities that may help systems maintain value delivery in spite of change, [5] including reliability [6], security [7], safety [8], survivability [9], resilience [10-12] and robustness [13]. While the usefulness of these various "-ilities" has been established, many of them are similar and possibly redundant. Are all of these "-ilities" required to sustain value? How is resilience different from survivability? Is reliability necessary to achieve robustness? To answer these questions, this section compares the similarities, differences, and gaps in some of the more common "-ilities" in the literature to determine the quality or qualities that a system needs to possess to maintain its value delivery in spite of change.

A. Resilience

Out of all of the "-ilities", the term "resilience" is probably used the most often to describe how a system maintains stakeholder value over time. The Oxford dictionary [14] defines resilience as "the capacity to recover quickly from difficulties" and "the ability of a substance or object to spring back into shape." Unfortunately, the definition of resilience varies widely in the engineering literature. Sheard & Mostashari [15] list 35 different definitions of resilience, highlighting the fact that they mainly differ along five dimensions: (1) the period of time over which the perturbation occurs, (2) the type of system, (3) the events that occur, (4) the actions the system must take, and (5) the qualities that the system is trying to preserve. There is no consensus in the literature as to what a "resilient" system actually is, as for example, certain definitions might require that a system change (or at least have the ability to change), while other definitions explicitly require that the system stays the same when everything else changes. Due to the ambiguity of the definition, it is not advised to require that a system be "resilient" without explicitly stating exactly what that entails.

B. Reliability

The IEEE [16] defines "reliability" as the ability of a system or component to perform its required functions under stated conditions (i.e. context and needs) for a specified period of time. Reliability ensures that the system itself doesn't change over time in such a way that it can no longer provide stakeholder value. Reliability is necessary for engineered systems to provide value to stakeholders over its expected lifetime, however it is not sufficient if those systems are subject to contextual perturbations outside what should be expected. For example, an AV would be considered unreliable if all the power supplies fail and the AV cannot operate. It would also be considered unreliable if the communication system failed because rain interfered with the signal propagation through the air. If, however, the AV failed because it was shot down by an enemy missile, then the AV would not be considered unreliable. To enemy attacks and other unusual perturbations, other "-ilities", such as survivability and robustness, need to be examined.

C. Survivability

Ellison et al. [17] has defined survivability as "the capability of the system to achieve its requirements or goals, in a timely manner, in the presence of attacks, failures or accidents." Richards [18] has defined survivability as "the ability of systems to minimize the impact of finite-duration, contextual changes on value delivery," and introduced two new survivability metrics: (1) time-weighted average utility loss, and (2) threshold availability. Typically, there are three ways systems can achieve survivability [10]: (1) reducing the probability that a disturbance will impact the system, (2) reducing the amount of value lost directly as a result of a disturbance occurring, and (3) increasing the system's ability to make a timely recovery from a disturbance, known as system resilience. The three ways to achieve survivability are also referred to as Type I, Type II and Type III survivability, respectively [9]. Jackson [12] refers to the same concepts as Richards, but uses the term "resilience" instead of "survivability"[1].

It is important to note that survivability is relative to a specific event or set of conditions. We can say that an engineered system is (or is not) survivable to *X*, where *X* may be some unusual event such as a thunderstorm, collision or hacker attacks. It is not meaningful to say that a system is survivable (in general), mostly because it is impossible to enumerate all possible disturbances a system may encounter during its lifespan.

D. Robustness

Robustness is another quality that is closely related to survivability. The Oxford dictionary [19] defines robustness of a system or organization, as "being able to withstand or overcome adverse conditions," but like the term "resilience," there are several different definitions in the technical literature. The "Taguchi" method, popular in industrial

[1] Incidentally, Richards uses the term "resilience" to refer specifically to Type III "survivability", which Jackson calls "recovery."

engineering for increasing product quality, defines a robust design as one which delivers a strong "signal" regardless of external or internal "noise" [20]. The concept of "signal" and "noise" relates to what the design is trying to do (the "signal") and external and internal disturbances that interfere with that ("noise"). During a workshop on systems engineering, Ross, Rhodes and Hastings [21] noted that Dr. Marvin Sambur, then Assistant Secretary of the Air Force for Acquisition, defined system "robustness" as being adaptable, scalable, reliable, sustainable, and easily modifiable, which is somewhat ambiguous. For example, what does "easily modifiable" mean? Does it mean that it is "easy" to modify because it is inexpensive, or is it "easily modifiable" because changes do not require much time or effort? Are changes that have to be made by external entities still considered "easy" or do the changes have to be endogenous?

Beesemeyer [22] clarified the definition of robustness, while distinguishing it from similarly defined "-ilities." According to Beesemeyer, a system *is* something (e.g. form), and *does* something (e.g. function), which can be described by a set of system parameters and outcome parameters, respectively. For example, an AV can be described by a set of system parameters, such as the fact that it is 5.4 m long, and output parameters, such as the fact that it can fly at 120 km / h. Robustness can be defined as a system's ability to maintain its output, regardless of changes in the system or context. If AV could still fly at 150 km /h even if the diesel engine of the AV was replaced with a solar-powered electric motor or if the AV flies into a hurricane, then the AV would be considered (at least partially) robust in air speed to changes in engine technology and weather conditions. Robustness makes no claim about value delivery, and the distinction between output and value is important. If the stakeholders of the AV decide in the future that the AV needs to fly at 200 km / h and the AV cannot do so, then the AV is not valuable (i.e. viable), even if it is still robust. This distinction implies that while robustness is a useful quality to have, it is not sufficient to ensure a system will maintain its value delivery over its lifecycle.

E. Value Sustainment

Due to the ambiguity and lack of consensus on terms like robustness in the literature, Ross and Rhodes [23] define *value robustness* as "the ability of a system to continue to deliver stakeholder value in the face of changing contexts and needs," which may be achieved through various related "-ilities," such as modifiability, adaptability, scalability, and flexibility. The concept of value robustness is typically used to describe the ability of a system to maintain value delivery in response to variations within considered contexts. These variations are sometimes referred to as "long term context perturbations." The concept of epoch, which is a time period of fixed context and needs, can also be used to consider exogenous factor variations that might impact a system [24]. Enumerating sets of possible epochs, and evaluating system sensitivity to variation in experienced epochs is another technique for evaluating system robustness. Generalizing this concept, Beesemeyer [22] defines "value sustainment" as the ability to maintain value delivery in spite of long or short-term contextual perturbations, which include both disturbances and context changes (i.e. epoch shifts).

Value sustainment is the closest "-ility" that would describe the quality that system architects are looking for, however it is only useful for situations where there may be contextual changes that the system needs to be able to handle. For most complex systems, such as an AV, this is typically the case. However, there are situations where engineered systems are expected to operate in static contextual conditions and therefore do not need to be value sustainable. For example, an inexpensive stereo is an engineered system that is expected to operate for years in a relatively stable context. If it meets the needs of the buyer initially, then it simply needs to be reliable for the rest of its lifetime. Unless the stereo was very expensive, it is not usually a requirement for it to be changeable, nor survivable to external disturbances such as power surge. If the owner's needs change over time, then he or she will simply replace the stereo. While the case where the context remains static may seem idealistic and not applicable for complex systems, a strategy for ensuring that a system maintains its value may be to place it within a system of systems (SoS) where it can be shielded from external perturbations and therefore operate under a pseudo-static context. For example, a particular AV may not be survivable to missile attacks, however it could be viable if it is located within a SoS where other SoS components can intercept missiles before they reach the vulnerable AV. In this way, the local context of the AV does not include missile attacks, even though the overall context for the SoS does.

III. Viability for Engineered Systems

Although many of the -ilities" described in the previous section are useful for helping systems maintain their value delivery under certain circumstances, the definition for viability incorporates several important concepts, which together form a unique, but important property in the technical literature. These concepts are now enumerated.

1) Viability is applicable to all engineered systems.

The concept of viability is applicable to all engineered systems, whether they are traditional, monolithic systems, or large systems of systems. The concept of viability is not intended for other systems, however, such as natural systems. Other definitions of viability, such as those that use the VSM, may be more appropriate in those cases.

2) Viability is subjective.

Whether a system is viable or not, is determined by how well the outputs of the system are likely to satisfy stakeholder needs. Different stakeholders have different preferences, so the same system could be viable in the same context for one stakeholder, but not viable for another, simply because the stakeholder needs are different.

3) Viability is dynamic.

Viability is a prediction about whether the system will

provide acceptable value to its stakeholders over its *life era.* What constitutes the life era is a prediction made by the stakeholders at the time viability is assessed. At the design stage, the life era may be determined to be a particular set of epochs, but as the system is implemented and exists, its life era (or what is left of it) may start to change, as the context and/or stakeholder needs change. As a result, the viability of a system may change as the system or context changes.

4) Viability is relative.

Although this research does not attempt to define metrics for viability, a system can be more or less viable than another system, or to itself if something changes, since viability is a likelihood. The more likely that a system will provide acceptable value to its stakeholders over its life era, the more viable it is. Viability can be *enabled* by something, if that something increases the likelihood that a system will provide value to its stakeholders over its life era.

5) Viability is not existence.

It is possible for an engineered system to exist, for a finite period of time, without being viable. This is because the system was viable when implemented, but something changed and caused it to become not viable, even though it still exists. An example of the difference between existing and viability would be if an AV was built to perform surveillance and did so for a few years. Then, a change in environmental regulations is announced that requires changing the diesel engine to something more environmentally friendly. Stakeholders balk at the cost and the system becomes not viable, even though it still exists and may even provide acceptable value until the law takes effect.

6) Viability is a prediction.

Viability is a prediction about whether the system will continue to provide value to its stakeholders over its life era, which may involve a considerable amount of uncertainty and risk. If the system is complex, there may be some uncertainty about its outputs. Similarly, there may be additional uncertainty about the context and needs of an engineered system that is expected to operate in a dynamic environment over a long time period. When both of these sources of uncertainty are taken together, it can make assessing viability very difficult [25].

7) Viability is not invulnerability.

A system can be viable, even in a context where there are threats and perturbations that it may not be able to survive, as long as the stakeholders are willing to accept the risk. For example, an AV could be considered viable, even if it could be destroyed by a nuclear missile, as long as the stakeholders are willing to exclude nuclear missiles from the list of contextual perturbations being considered. The Ford Pinto fuel tank controversy [26], is an infamous case where Ford executives were accused of approving a flawed design, after performing a cost-benefit analysis that included legal damages resulting from deaths. This type of risk-taking may also highlight the fact that viability of a mass-produced system, such as a car or cell phone, is likely to be different than the viability of a massive, unique SoS like the Next Generation Air Transportation System (NextGen) due to the risks of failure. In mass-produced systems, a certain number of systems are often expected to fail, with relatively small consequence. In larger, especially unique systems, failure of a system is catastrophic to its stakeholders.

8) Viability does not require independence.

According to the Viable System Model (VSM), a system must retain its independent existence in order for it to be viable [27]. Independence, however, is subject to interpretation. Most, if not all, systems require certain contextual conditions to exist and some of these conditions include the presence of, and actions by, certain entities. For example, businesses without clients or customers would not be able to exist, and therefore would not be viable. The VSM handles this by assuming that adequate resources and services are part of the environment, at least to some extent. A viable system, according to cybernetic management, can handle perturbations in the environment, such as a reduction in available resources, but it cannot be viable if critical resources disappear entirely. If some of these critical resources are in fact other systems, then are the organizations described by the VSM independent? It depends on how independence is defined. In terms of engineered systems, there are ten levels of automation, ranging from a "dumb" Level 1 system where the human does all the decisions and actions, to a fully automated Level 10 system that has no human in the loop whatsoever [28]. The fact that Level 1 engineered systems exist and provide acceptable value to stakeholders, show that engineered systems can be viable without being independent. For this reason, independence of components will not be a requirement for viability of an engineered system.

9) Viability does not require the ability to change.

Another requirement for a system to be viable, according to the VSM, is that it has the ability to change, or adapt, in response to changes in the environment. For a complex AV like a Global Hawk, which is expected to take part in numerous different missions, in dynamic environments, within and without a SoS, it is hard to imagine it being viable without the ability to adapt in some way to meet emerging technological and social changes. However, for simpler engineered systems such as a home stereo, it is reasonable to expect the context to remain fairly static. Thus, the requirement for change or adaption is not necessary for viability in engineered systems.

IV. Designing for Viability

If a system architect wishes to design a system that provides value over its entire lifetime in spite of perturbations that may arise, then the system will need to satisfy two criteria. First, the system must survive any inevitable events that threaten the value delivery of the system. Second, the system must prevent terminal events from occurring. A terminal event is any event

that the system cannot successfully mitigate or recover from [25]. If the system cannot satisfy both criteria, then it is not viable.

Due to events that are beyond the control of a system or its stakeholders, a viable system will need to be survivable to certain perturbations that arise during its lifetime. This means that the numerous survivability design principles already in existence, such as Richards [29] and Jackson [12], will also be effective viability design principles for relevant systems and disturbances. However, it is possible that a system may not need to be survivable to any perturbations and still be viable, if those perturbations do not occur. A system is fragile to a particular context if its viability is dependent on the context remaining static. A fragile system does not need to be survivable or value robust, since those properties are defined by the ability to maintain value in spite of change. Shielding the system from contextual changes is a strategy for achieving viability [24], much in the same way a mother shields a newborn from perturbations it cannot survive on its own.

V. System Change as a Disturbance

For systems that cannot be shielded from contextual perturbations, researchers have studied designing latent capabilities that grant a system the ability to change. This includes changing at a later date [30], changing during operational phases [31], changing easily / rapidly [32], and changing in size [33], just to name a few examples. Being able to change, known as changeability [21] is just one of the "-ilities" that addresses the need for a system to change in different ways, to respond to various types of contextual perturbations. Other change-related "ilities" include agility, flexibility and evolvability, as shown in Table I [34]. Researchers have shown that change-related "-ilities are desirable properties to have, but is changeability always a good thing? Does making a system more changeable improve its ability to deliver value to its stakeholders? Can changeability actually decrease value delivery to stakeholders and become a vulnerability?

TABLE I
"-ilities" Relating to Change [34]

"Ility"	Description
Adaptability	Ability to be changed by a system-internal change agent with intent
Agility	Ability to change in a timely fashion
Changeability	Ability to alter its operations or form, and consequently possibly its function, at an acceptable level of resources
Evolvability	Ability to be inherited and changed across generations (over time)
Extensibility	Ability to accommodate new features after design
Flexibility	Ability to be changed by a system-external change agent with intent
Interoperability	Ability to effectively interact with other systems
Modifiability	Ability to change the current set of specified system parameters
Modularity	The degree to which a system is composed of modules (not an ability-type ility)
Reconfigurability	Ability to change its component arrangement and links reversibly
Robustness	Ability to maintain its level and/or set of specified parameters in the context of changing system external and internal forces
Scalability	Ability to change the current level of a specified system parameter
Survivability	Ability to minimize the impact of a finite duration disturbance on value delivery
Value Robustness	Ability to maintain value delivery in spite of changes in needs or context
Versatility	Ability to satisfy diverse needs for the system without having to change form (measure of latent value)

A. Dispatch Failure

One of the most well-known examples of an endogenous change gone wrong [35] leads credence to the popular saying - "If it isn't broke, don't try to fix it." The London Ambulance System (LAS) is the largest free to patient ambulance services in the United Kingdom, responding to over 1.5 million calls a year in the London Area. On October 29, 1992, a new Computer Aided Dispatch (CAD) was introduced into the existing ambulance network to optimize ambulance allocation and reduce wait times. Overloaded with calls and working with imperfect information, the new CAD failed and the LAS was unable to respond to emergency calls and send ambulances in a timely manner. As a result, the LAS was blamed by the media for as many as 30 deaths [36]. After only a few days in service, the CAD was removed. According to the official report, there were numerous problems that led to the LAS failure that occurred, including organizational resistance (i.e. some ambulance operators were deliberately ignoring the new operating procedure) and project management. A fundamental problem, however, was that the new software caused an unnecessary bottleneck, by making some wrong assumptions about how the existing system operated.

B. The Cluster

Another infamous example of endogenous change failure occurred in 1996 when the Ariane 5 rocket failed to achieve orbit, losing the European Space Agency Cluster spacecraft it was carrying. This $370 million [37] accident occurred because of a bug in a particular software component that caused a hardware failure elsewhere in the system. As Nuseibeh [38] explains, the real problem was that there was a risk in one of the components that increased as requirements were changed. Although there was a threat, the original design was viable because the stakeholders accepted the risk of not handling a certain software exception. However, as the system changed to meet new requirements, that particular risk increased. Project managers failed to realize this change in risk, however, and ended up reusing the old code within the new design, with disastrous results.

For both the LAS and the Cluster failures, voluntary endogenous changes that were made to improve value delivery, actually ended up doing the opposite. Clearly, a changeable system may be desirable as research has shown, but is it always better than a non-changeable system? What changes should and should not be allowed? To answer these questions, we next define what constitutes a system change.

VI. System Architecture and Change

Changes can be described in many dimensions including the time it takes to make the change, the cost of the change, the complexity of the change, the location of the agent performing the change (i.e. exogenous or endogenous to the system) and whether the change was intended or not. Ideally, an intentional change should have predictable consequences. However, many changes, particularly those performed on complex systems such as an AV, will have unintended consequences as well. Predicting all of the effects that an endogenous change has on the system itself often requires an in-depth understanding of the system architecture.

The Department of Defense defines system architecture as a framework or structure that portrays relationships among all the elements of the system [39]. Crawley [40] defines system architecture as the description of the components of a system and the relationships between them. In this sense, the system architecture includes both what the system is composed of, and how these components work together. At a minimum, the system architecture should describe the functional behavior of the system, i.e. which tasks the components perform, as well as when and how the tasks are executed, similar to the Capability and Operational Views found in the Department of Defense Architectural Framework 2.0 [41]. We define components to be what a system is supposed to be made up of, and a Concept of Operations (CONOPS) as what a system is supposed to do. The components of a system includes all of the possible entities that make up the system, while the CONOPs describe how those entities interact with each other and the environment to deliver value to the stakeholders. Together, the components and the CONOPs of a system describe the system's architecture. Although a system's components and CONOPs can be changed independently, often a change in one requires a change in the other. A change in either components or CONOPs means that the system itself has changed.

For simple systems, the emergent behavior of the system is often predictable from the system architecture, through aggregating the functional behaviors of the components. For more complex systems, emergent behavior is more difficult to predict and the primary technical role and responsibility of a system architect is to ensure that the interconnections of the components achieve the necessary "system functions" to meet the overall purpose [42].

A simple approach to change would be to categorize them as being either intentional or unintentional, depending upon whether the decision makers in charge of the system made the decision to change or not. Intentional changes are often in response to a shift in context or stakeholder needs, and are typically the types of changes described by researchers when they discuss changeability, flexibility, agility and other types of change-related system properties. Unintentional, or "imposed," changes are those that the system is "forced" to undergo, that is, changes that occur whether the decision makers want them to or not. Some unintentional changes occur spontaneously, such as when a collision physically alters the structure of a system that has been impacted, while others are unintentional endogenous changes that are in response to an exogenous disturbance. Other unintentional changes happen after decision makers authorize them, but only because they have to, in response to some other event beyond their control. A labor strike may shut down an AV, or a new regulation may require that the AV incorporate new sense-and-avoid technology.

Much of the work in resilience engineering is really about making sure that a system resists certain unintentional changes. However, the concept of "intentional" changes should be clarified further than has traditionally been the case. Sometimes, decision makers may want to implement a change to improve value delivery, but end up reducing it instead. This is because as the system complexity grows, and as more changes are made to the system, it becomes increasingly difficult to verify and validate the effects that changes will have. Sometimes, the original architects and engineers who designed the system and really understood how it works may no longer be available to evaluate impacts of planned changes. Subtle assumptions that were not made explicit may be violated with new changes and could prove to be not only disastrous, but also difficult to find until it is too late. The larger and more complex a system is, the more likely this will be the case. This is particularly a problem with systems of systems that have autonomous constituent systems with emergent behavior that is difficult to model. The larger the system, or the more expensive, or the more stakeholders involved, the likelihood decreases of someone wanting to accept responsibility for any one particular change. Thus, there may be a substantial bureaucratic process involved for any complex system that makes it extremely time-consuming and costly for all but trivial changes to be made. This "red tape" can seriously impair the ability of the system to respond quickly to changes in context, especially if it takes years to approve any significant changes.

VII. Change Within and Between Architectures

To help determine whether a change might threaten the value delivery of a system requires understanding how systems can transition within and between system architectures. This requires defining the concept of an architectural instance. A typical system architecture will have fixed and variable system parameters in the components, CONOPs or both. The fixed system parameters differentiate one architecture from another, while the variable system parameters differentiate one architectural instance from another. A design is a specific set of components (known as its form) and a specific CONOPs (known as its mode of operation) that belongs to a system architecture. Thus,

$$D_{ij} = [F_i, MO_j] \quad (1)$$

where D_{ij} is the design consisting of the *ith* form specified in the set of components and the *jth* mode of operations specified in the set of CONOPs of some particular system architecture. A design is conceptual and it only exists as an idea. However,

a *system* is an actual physical realization of a design that provides value to stakeholders. A design is an instance of an architecture, and a system is an instance of a design.

To illustrate the differences between various system architectures and designs, consider designing an AV to conduct surveillance missions. In reality there will be numerous parameters to specify, but for illustrative purposes, only the following four binary parameters are relevant:

- Rotary or fixed wing-type [form]
- Electro-Optical (EO) or Infrared (IR) payload [form]
- Sense & Avoid (S&A) technology enabled or not [form]
- Solo or SoS flying formation [mode of operation]

A. Feasible & Viable Designs.

Table II enumerates all 16 possible designs, but some of them are not feasible. Suppose there is a stakeholder constraint that requires all unmanned aerial vehicles to have S&A technology in order to fly in a SoS formation. This limits the number of feasible designs down to 12. Further suppose that the system analysts determined that rotary wing designs were vulnerable to missile attacks, due to their slower speed and lower altitude ceiling. While those designs are feasible, they are not viable in a war context. However, when part of the SoS which includes anti-missile technology, the local context of rotary-wing AVs does not include missiles, and therefore designs that fly within the SoS are viable.

TABLE II
FEASIBILITY AND VIABILITY OF VARIOUS SYSTEM DESIGNS

DESIGN	WING	PAYLOAD	S&A	MODE	VIABILITY
D_{F1}	Fixed	EO	No	Solo	Viable
D_{F2}	Fixed	EO	No	SoS	Not feasible
D_{F3}	Fixed	EO	Yes	Solo	Viable
D_{F4}	Fixed	EO	Yes	SoS	Viable
D_{F5}	Fixed	IR	No	Solo	Viable
D_{F6}	Fixed	IR	No	SoS	Not feasible
D_{F7}	Fixed	IR	Yes	Solo	Viable
D_{F8}	Fixed	IR	Yes	SoS	Viable
D_{R1}	Rotary	EO	No	Solo	Not viable
D_{R2}	Rotary	EO	No	SoS	Not feasible
D_{R3}	Rotary	EO	Yes	Solo	Not viable
D_{R4}	Rotary	EO	Yes	SoS	Viable
D_{R5}	Rotary	IR	No	Solo	Not viable
D_{R6}	Rotary	IR	No	SoS	Not feasible
D_{R7}	Rotary	IR	Yes	Solo	Not viable
D_{R8}	Rotary	IR	Yes	SoS	Viable

VIII. PLIABILITY

Pliability is the property of a system to be able to switch to other viable instances, specified by the architecture's pliable set. The pliable set is the set of allowable designs that adhere to the architecture, have been validated (by whatever means the architect desires) to deliver acceptable stakeholder value, and are connected (i.e. can transition to, or be transitioned from another viable instance within the system architecture). If a system architecture has multiple allowable designs, then a system is always an instance of one of these designs at any time *t*, and can transition to the other instances defined in the pliable set of its system architecture, while remaining the same system. If a system transitions to an instance outside of its system architecture, then it becomes an unapproved system. Whether this new system will be viable or not is unknown (at best), since it does not belong to the set of viable instances, defined by the architects responsible for its value delivery. Thus, it is usually in the best interest of the architects and decision makers to not let systems change into instances outside of their system architecture.

A. Pliable Transitions

A system's ability to transition to other systems is referred to as system changeability [24]. Changeability is affected by a number of factors including stakeholder support and cost. An allowable transition is one where the switch from one design to another is allowed by the system architects. There are a number of reasons why a system transition may not be allowed by the system architects. The question of whether system architects would allow a transition would likely depend on the answer to the following questions: Is there enough time to transition? Is the cost of transition worth it? Does it matter if none of the original components are used in the final design (i.e. they all get removed or replaced at some stage in the transition)? Does it matter whether or not the intermediate stages between the original stage and the final stage are still viable systems themselves?

Although there are eight viable designs in the AV example above, not all designs may be reachable from all other designs. Determining which designs are reachable from other designs is how the system architects define the pliable sets and the system architectures. Suppose there are two constraints on transitioning between designs. The first constraint is that the choice of wing-type is fixed, meaning that a system that is fixed-wing cannot be transformed into a rotary-wing design and vice-versa. The second constraint, is that while the S&A technology can be added at any time to a design, the installation is irreversible, meaning that the transition is one-way only (from no S&A to having S&A).

Seeing as there are no constraints on transitioning between the two payloads or between the flying formation (other than the fact that SoS flying requires a S&A to be installed), the constraints divide the designs into two system architectures (defined by the wing-type) and three pliable sets, as shown in Fig. 1. These are pliable set I = (D_{F1}, D_{F3}, D_{F4}, D_{F5}, D_{F7}, D_{F8}), pliable set II = (D_{F3}, D_{F4}, D_{F7}, D_{F8}), and pliable set III = (D_{R4}, D_{R8}). Pliable sets I and II belong to one SA (the fixed-wing architecture), whereas pliable set III belongs to the other SA (the rotary-wing architecture).

B. Non-pliable Transitions

If a system changes in ways allowed by its system architecture, then this is a pliable change and something that is allowed, if the context change requires it. This is one of the main attractions of having a pliable system. However, if the new instance is not allowed by the system architecture, then this change can be considered a "hack." Sometimes hacks

work, such as when users "jailbreak" their Apple iPhone 4S to run on unauthorized networks. Other times, the hacks fail, due to the complexity of the system and the fact that the changes were never considered or approved by those who were responsible for building the system in the first place. In some cases, non-pliable changes can result in a system that becomes a design of a different system architecture, if such an architecture is already defined, or requires a new system architecture to be defined. This is a more serious change that often requires stakeholder approval and input from system architects. If the new system architecture is validated by the system architects, then, like pliability, this type of transition is a special case of changeability known as evolvability [43].

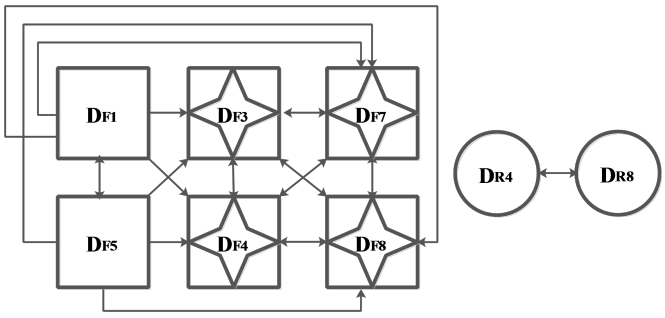

Fig. 1. Designs belonging to the three pliable sets (represented by the square, star and circle symbols).

Another type of system transition is degradation. This type of transition is when the system changes in a way that deviates from any design within the system architecture and fails to provide value accordingly. For example, losing the diesel engine would result in a degradation, since all designs require it. For the system to return to viability, it would need to replace the diesel engine and restore the system, or transition to some other design that is also viable but belongs to a different system architecture. Fig. 2 shows the types of transitions discussed in this section.

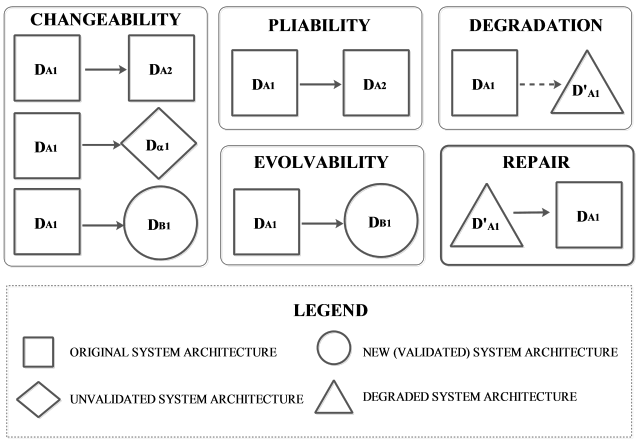

Fig. 2. Various System Transitions Within and Between Architectures.

IX. Pliability and Viable Systems

Viability requires the effective handling of change within systems, either by preventing, mitigating or recovering from unwanted change caused by perturbations, or intentionally changing in response to new contexts. By requiring that systems be pliable, viability can be increased in three different ways: (1) by placing constraints on transitions so that stakeholders do not decide to change the system into something that is not viable (2) by introducing viable instances that stakeholders may not have considered, so that system can transition if needed, and (3) by pre-validating change options to reduce the time and effort required for stakeholders to approve changes, allowing them to be implemented quicker and easier. The reason why pliable set I is distinct from pliable set II, is because certain designs in II can never reach certain designs in I, due to the S&A installation constraint. This is important, because it means that the designs in pliable set II have less viable designs to transition to, in the vent of a contextual change. Thus, the viability of a particular design is not only dependent on how well that design delivers value to its stakeholders, but also on whether or not that design can transition to another viable design, if needed.

X. Resilience of Pliable Systems

Disturbances may break the architecture of a particular system by altering one of the fixed parameters, or by forcing a parameter to go outside its pliable range. Since the system was not designed to operate outside of its architecture, there is a risk that the disturbance will not be survivable if the new, architecture non-conforming instance is not viable. In this situation, the system needs to either return to its original architecture through repair or replacement, or adapt to a new, viable system architecture (SA). In other cases, disturbances may not break a system architecture, but instead change the context in such a way that the system will not survive unless it adapts to a new, viable system architecture (Table III).

XI. Pliability Strategies for Enabling Viability

Using the concept of pliability, several new strategies for viability can be developed (Fig. 3). The strategies are Reversion, Stable Intermediate Instances, and Contingency. These strategies attempt to reduce the risk associated with making a large transition, by being able to transition into viable instances that are not necessarily the instances to which the system was intending to transition.

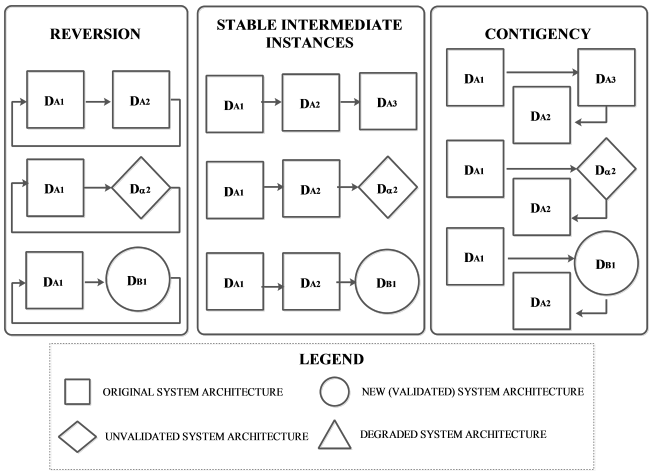

Fig. 3. Viability Strategies Using Pliability.

A. *Strategy of Stable Intermediate Instances*

This strategy is transitioning to a less-risky instance within the pliable set, before attempting to transition to a higher-risk instance, in case the system never makes the full transition. This strategy is often used with Reversion. For example, suppose stakeholders of an AV described in Section VII that was fixed-wing with no S&A technology wanted to go beyond the pliable set and include the ability to carry dangerous cargo, as well as incorporating S&A technology. Both of these changes are irreversible, so reversion to the original instance is not possible. Instead of transitioning directly to an invalidated design by adding the cargo capacity first, a better strategy might be to first transition to a stable design by incorporating the S&A before the cargo. This way the stakeholders have another chance to decide if the final version is really want they want before making the irreversible addition. If the context is dynamic, it is possible that the stakeholders may not want or need to transition outside the system architecture, and the stable intermediate instance may be good enough.

B. *Strategy of Contingency*

Sometimes it is not practical or even possible to build a stable intermediate instance before transitioning to a final design. Contingency is when there is a stable design than can be reached from the final design that the system can transition to in the event that it needs to (i.e. a "fall back" option).

XII. Pliability and other Change-Related "ilities"

At first glance, pliability may seem to be the same as some other "ilities" already in use, such as "flexibility", "modifiability" or "adaptability." The following is a description of how pliability is related to, but distinct from, the other "ilities" as defined in Table I.

A. *Value Robustness*

Pliability is only concerned with changes in the system, and whether such changes are allowed under the SA. Implicit in the specification of "allowable under the SA" is the concept of value robustness. Pliable systems that undergo an allowable transition will retain value robustness for contexts that were considered in the design of the SA. Pliability does not guarantee value robustness in contexts that were not considered in the design of the SA, regardless of whether the system transition was allowable or not.

B. *Modifiability*

Pliability requires either modifiability or scalability, i.e. the parameter can change in some way. However, a system may be modifiable in a parameter beyond what is included in the pliable range.

C. *Changeability*

A system must be changeable to be pliable, but does not have to be pliable to be changeable. Pliability requires not only that the transition to a new system is possible (i.e. the system is changeable), but also that the new system is part of the same system architecture (not needed for changeability). In this way, changeability is more general than pliability.

D. *Adaptability*

Pliable systems need to be changeable, but they do not need to be changeable by an internal change agent. A system that is changeable only by an external agent can also be pliable.

E. *Flexibility*

Pliable systems need to be changeable, but they do not need to be changeable by an external change agent (an internal change agent will do), nor does the agent need intent. Pliable systems can transition to other systems by accident (e.g. as the result of a disturbance). In fact, one of the main goals of having a pliable system is that if a disturbance does impose an involuntary transition, the transition will be more likely to result in a system still within the system architecture.

F. *Evolvability*

A system that transitions to another instance within its SA is demonstrating pliability not evolvability. Evolvability requires changes in a system with inheritance during "redesign." This may correspond to changes in an inherited system architecture.

G. *Agility*

Pliability does not explicitly require timely changes, just as long as the transition "is possible." If stakeholders are not concerned with timeliness, then timeliness is not a factor in the pliability of the system. However, pliability may enable agility by reducing "red tape" for change.

H. *Scalability*

Pliability requires either modifiability or scalability, i.e. the parameter can change in some way. However, a system may be scalable in a parameter beyond what is included in the pliable range.

I. *Versatility*

Pliability allows change in form and/or mode of operations. A system may be versatile and pliable if it changes CONOPs (only), within the pliable range, to satisfy diverse needs.

J. Reconfigurability

Reconfigurability is a specific change in CONOPS (one that changes the relationships between the Operational Elements). This change may or may not be in the pliable range, hence a system may be pliable and reconfigurable, pliable only (if the allowable changes do not include CONOPs changes concerning component relationships) or reconfigurable only (if the CONOPs changes are not part of the pliable range).

TABLE III
RECOMMENDED ACTIONS BASED OF EFFECTS OF DISTURBANCE

Effect of Disturbance	Recommended Action
Changes a system's parameters that violate the pliability of SA	Repair system to original SA, validate new SA, or change to an alternative, viable SA
Changes context so current SA not producing value	Change SA
Change context so current system not producing value	Change system instance

K. Extensibility

Assuming that "new features" are features not included in the original design (i.e. system architecture), then extensibility and pliability are independent properties. For example, an AV can be both extensible and pliable. Performing an authorized upgrade from standard definition video payloads to high definition video payloads is an example of pliability, if both payloads are part of the system architecture for that AV. Adding third-party peripherals or software that is not part of the system architecture is extensibility. An example would be if an additional communications relay payload was added to an AV that did not include that feature as part of its system architecture. Like any other change, adding features to a system may violate the architecture, if it causes a parameter to change outside its pliable range (e.g. "power supply needed"). Architects should try to safeguard the parameters of their system architecture when designing for extensibility.

L. Robustness

A pliable system may be robust if it does not need to transition to a new instance to provide value under a new context. Likewise, robustness can be achieved by transitioning to a new instance within the pliable set as a response to perturbations to maintain outcome parameters. Pliability provides the system with the option to change, but does not require it to.

M. Modularity

Pliability is not concerned with the ability of a system to be composed of modules, although, modularity will likely make a system more changeable, thus facilitating pliability should the system architects wish to make those parameters pliable.

N. Interoperability

Not directly related to pliability, but a system architecture that supports interoperability will likely require a larger pliable set than a similar architecture that does not, particularly if the system is taking a defensive posture (proposed Type I survivability design principle). This is because the range of what the system may need to accommodate will likely be larger than what would be needed if the interfaces were custom.

O. Survivability

Systems that are more pliable will be more survivable, due to the fact that there are validated system changes available should stakeholders need to change in response to a disturbance (Type II survivability). Systems may also be more survivable due to an increase in the likelihood that an involuntary change brought about by a disturbance will still transition into one of the validated instances of the system architecture (Type II survivability).

XIII. CONCLUSIONS

When complex, engineered systems are expected to provide value to stakeholders over long periods of time and under hostile or dynamic contexts, they need to be more than just feasible – they need to be viable. In this paper, viability for engineered systems was defined and differentiated from other similar, existing "-ilities. In order for an engineered system to be viable, it must be able to avoid perturbations that it cannot survive, and survive perturbations it cannot avoid. While many of the existing survivability strategies are still effective at enabling viability, different strategies emerge as a result of this new construct. For example, an effective viability strategy for avoiding contextual perturbations is to keep the system's local context constant by shielding the system from the real context in which it operates. This can be done in systems of systems, where the local context of constituent systems is still within the overall SoS.

Viability is not only limited to contextual considerations, but also to managing endogenous change as well. It was demonstrated that while a considerable amount of research has been done to promote intentional change as a solution for contextual perturbations (i.e. flexibility), some restraint should be exercised. Endogenous change that ignores or violates how the system is supposed to operate (i.e. the CONOPs) runs the risk of being a disruptive disturbance itself, with unintended consequences that may threaten the viability of the system. To help system architects define where change should and should not be allowed, and to increase the likelihood of allowable changes made (i.e. reducing the red tape and increasing the agility of the system), the concept of pliability was introduced. Like viability, pliability is a newly defined construct in the technical literature, and was shown to be distinctly different from other similar change-related "-ilities." Through the example of an AV system, pliability was demonstrated as a useful architecting construct for increasing the viability of engineered systems by pre-validating system transitions that will benefit the system, and restricting those that will not.

REFERENCES

[1] B. Mekdeci, A. M. Ross, D. H. Rhodes, and D. E. Hastings, "A taxonomy of perturbations: Determining the ways that systems lose value," in *2012 IEEE International Systems Conference*, Vancouver, Canada, 2012, pp. 1-6.

[2] A. Dijkstra, "Cybernetics and Resilience Engineering: Can Cybernetics and the Viable System Model Advance Resilience Engineering?," in *Resilience Engineering Workshop*, 2007, p. 23.

[3] S. Beer, "The viable system model: its provenance, development, methodology and pathology," *Journal of the operational research society,* pp. 7-25, 1984.

[4] W. R. Ashby, "Principles of the self-organizing system," *Principles of Self-organization,* pp. 255-278, 1962.

[5] H. McManus and D. Hastings, "A framework for understanding uncertainty and its mitigation and exploitation in complex systems," *IEEE Engineering Management Review,* vol. 34, pp. 81-94, 2006.

[6] J. Downer, "When failure is an option: redundancy, reliability and regulation in complex technical systems," London School of Economics and Political Science, London, UK.2009.

[7] C. M. Furlani, *Minimum Security Requirements for Federal Information and Information Systems*: DIANE Publishing, 2009.

[8] N. G. Leveson. (2002, Dec 11, 2012). *System Safety Engineering: Back to the Future*. Available: www.sunnyday.mit.edu/book2.pdf

[9] M. G. Richards, D. Hastings, D. Rhodes, and A. Weigel, "Defining Survivability for Engineering Systems," in *5th Conference on Systems Engineering Research*, Hoboken, NJ, 2007.

[10] R. Westrum, *A Typology of Resilience Situations*. Aldershot, England: Ashgate, 2006.

[11] E. Hollnagel, D. D. Woods, and N. Leveson, *Resilience Engineering: Concepts and Precepts*. Aldershot, UK: Ashgate, 2006.

[12] S. Jackson, *Architecting Resilient Systems: Accident Avoidance and Survival and Recovery from Disruptions*. Hoboken, NJ: John Wiley & Sons, 2010.

[13] A. G. Smart, L. A. N. Amaral, and J. M. Ottino, "Cascading failure and robustness in metabolic networks," *Proceedings of the National Academy of Sciences,* vol. 105, p. 13223, 2008.

[14] Oxford English Dictionary, "Resilience," in *Oxford English Language*, ed. Oxford, UK: Oxford University Press, 2012.

[15] S. Sheard and A. Mostashari, "A Framework for System Resilience Discussions," in *18th Annual International INCOSE Symposium*, 2008.

[16] Institute of Electrical and Electronics Engineers, *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. New York: Institute of Electrical and Electronics Engineers 1990.

[17] R. Ellison, D. Fisher, R. Linger, H. Lipson, and T. Longstaff, "Survivable network systems: An emerging discipline," Carnegie Mellon, PittsburghMarch 1998 1997.

[18] M. G. Richards, "Multi-Attribute Tradespace Exploration for Survivability," PhD, Engineering Systems Division, Massachusetts Institute of Technology, Cambridge, MA, 2009.

[19] Oxford English Dictionary, "Robustness," in *Oxford English Dictionary*, ed. Oxford, UK: Oxford University Press, 2012.

[20] G. Taguchi and V. Cariapa, "Taguchi on robust technology development," *Journal of pressure vessel technology,* vol. 115, p. 336, 1993.

[21] A. M. Ross, D. H. Rhodes, and D. E. Hastings, "Defining changeability: Reconciling flexibility, adaptability, scalability, modifiability, and robustness for maintaining system lifecycle value," *Systems Engineering,* vol. 11, pp. 246-262, 2008.

[22] J. C. Beesemyer, "Empirically Characterizing Evolvability and Changeability in Engineering Systems," Master of Science, Aeronautics and Astronautics, MIT, Cambridge, MA, 2012.

[23] A. M. Ross and D. H. Rhodes, "Architecting Systems for Value Robustness: Research Motivations and Progress," presented at the 2008 2nd Annual IEEE Systems Conference, 2008.

[24] A. M. Ross, "Managing unarticulated value: Changeability in multi-attribute tradespace exploration," PhD, Engineering Systems Division, Massachusetts Institute of Technology, Cambridge, 2006.

[25] B. Mekdeci, "Managing the Impact of Change Through Survivability and Pliability to Achieve Viable Systems of Systems," Ph.D., Engineering Systems, MIT, Cambridge, MA, 2013.

[26] M. T. Lee, "The Ford Pinto case and the development of auto safety regulations, 1893-1978," *Business and Economic History,* vol. 27, pp. 390-401, 1998.

[27] S. Hildbrand and S. Bodhanya, "The Viable System Model (VSM) and Qualitative Studies: A Research Perspective to Manage in a World of Complexity," in *10th European Conference on Research Methodology for Business and Management Studies*, 2011, p. 241.

[28] R. Parasuraman, T. B. Sheridan, and C. D. Wickens, "A model for types and levels of human interaction with automation," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on,* vol. 30, pp. 286-297, 2000.

[29] M. G. Richards, A. Ross, D. Hastings, and D. Rhodes, "Survivability Design Principles for Enhanced Concept Generation and Evaluation," in *19th Annual INCOSE International Symposium*, Singapore, 2009.

[30] R. de Neufville and S. Scholtes, *Flexibility in Engineering Design*. Cambridge, MA: MIT Press, 2011.

[31] Y. P. Gupta and S. Goyal, "Flexibility of manufacturing systems: concepts and measurements," *European Journal of Operational Research,* vol. 43, pp. 119-135, 1989.

[32] R. E. McGaughey, "Internet technology: contributing to agility in the twenty-first century," *International Journal of Agile Management Systems,* vol. 1, pp. 7-13, 1999.

[33] D. A. Elkins, N. Huang, and J. M. Alden, "Agile manufacturing systems in the automotive industry," *International Journal of Production Economics,* vol. 91, pp. 201-214, 2004.

[34] O. L. de Weck, A. M. Ross, and D. H. Rhodes, "Investigating Relationships and Semantic Sets amongst System Lifecycle Properties (Ilities)," presented at the 3rd International Engineering Systems Symposium CESUN 2012, TU Delft, 2012.

[35] P. Beynon-Davies, "Human error and information systems failure: the case of the London ambulance service computer-aided despatch system project," *Interacting with Computers,* vol. 11, pp. 699-720, 1999.

[36] T. Long. (2009, December 6, 2012). *This Day In Tech: Software Glitch Cripples Ambulance Service*. Available: http://www.wired.com/thisdayintech/tag/london-ambulance-failure/

[37] M. Dowson, "The Ariane 5 software failure," *SIGSOFT Softw. Eng. Notes,* vol. 22, p. 84, 1997.

[38] B. Nuseibeh. (1997, 1997) Ariane 5: Who Dunnit. *IEEE Software*. 15-16.

[39] Department of Defense. (2008, January 12, 2013). *Dictionary of Military and Associated Terms*. Available: http://www.dtic.mil/doctrine/new_pubs/jp1_02.pdf

[40] E. Crawley, O. de Weck, S. Eppinger, C. Magee, J. Moses, W. Seering, *et al.*, "The Influence of Architecture in Engineering Systems," presented at the MIT Engineering Systems Symposium, Cambridge, MA, 2004.

[41] Department of Defense. (2010). *DoD Architecture Framework 2.02*. Available: http://dodcio.defense.gov/Portals/0/Documents/DODAF/DoDAF_v2-02_web.pdf

[42] M. W. Maier and E. Rechtin, *The Art of Systems Architecting*: CRC Press, 2000.

[43] J. C. Beesemyer, A. M. Ross, and D. H. Rhodes, "An Empirical Investigation of System Changes to Frame Links between Design Decisions and Ilities," *Procedia Computer Science,* vol. 8, 2012.

Brian Mekdeci was born in Toronto, Canada and received a B.A.Sc. (2002) and a M.A.Sc. (2005) in systems design engineering from the University of Waterloo. In 2013, Brian earned a Ph.D. in engineering systems from MIT in Cambridge. He is currently a research fellow at the University of South Australia in the Defense and Systems Institute.

While earning his doctorate, he was a research associate at both the Systems Engineering Advancement Research Initiative (SEAri) and Humans and Automation Laboratory (HAL). Prior to enrolling at MIT, Dr. Mekdeci worked as a Systems Engineer at CDL Systems in Canada, where he was designing ground control software for unmanned aerial vehicles (UAVs). He also worked as a forensic engineering consultant, providing expert advice on a wide range of topics including forensic image processing and the effects of sleep deprivation on driver attentiveness. His research interests include survivability and resilience of systems of systems, unmanned systems, trusted autonomy, computer simulations, serious games and human factors.

Daniel E. Hastings Dr. Hastings earned a Ph.D. and an S.M, from MIT in Aeronautics and Astronautics in 1980 and 1978 respectively, and a B.A. in Mathematics from Oxford University in England in 1976. He joined the MIT faculty as an assistant professor in 1985, advancing to associate professor in 1988 and full professor in 1993. Dr. Hastings served ESD as the Director of the Technology and Policy Program from 2000-2003, Associate Director of ESD from July 2001 - April, 2003, Co-Director from May, 2003 - June, 2004, and Director from July 2004 - December 2005.

As Professor of Aeronautics and Astronautics and Engineering Systems, Hastings has taught courses and seminars in plasma physics, rocket propulsion, advanced space power and propulsion systems, aerospace policy, technology and policy, engineering education, and space systems engineering.

Dr. Hastings served as chief scientist to the U.S. Air Force from 1997 to 1999. In that role, he served as chief scientific adviser to the chief of staff and the secretary and provided assessments on a wide range of scientific and technical issues affecting the Air Force mission. He led several influential studies on where the Air Force should invest in space, global energy projection, and options for a science and technology workforce for the 21st century.

Adam M. Ross is a research scientist in the Engineering Systems Division at the Massachusetts Institute of Technology. He is co-founder and lead research scientist for MIT's Systems Engineering Advancement Research Initiative (SEAri), a research group focused on advancing the theories, methods, and effective practice of systems engineering applied to complex socio-technical systems through collaborative research with industry and government. Dr. Ross has professional experience working with government, industry, and academia. He holds a dual bachelor degree in Physics and Astrophysics from Harvard University, two masters degrees in Aerospace Engineering and Technology & Policy, as well as a doctoral degree in Engineering Systems from MIT.

Dr. Ross has research interests and advises ongoing research projects in advanced systems design and selection methods, tradespace exploration, managing unarticulated value, designing for changeability, value-based decision analysis, and systems-of-systems engineering. He is a leading thinker in designing for value robustness and tradespace exploration, developing several novel methods for quantitative analysis of alternatives across dynamic futures and multiple stakeholders. Dr. Ross has over 70 publications in the area of space systems design, systems engineering, socio-technical decision making, and tradespace exploration.

Donna E. Rhodes is a senior lecturer in the Engineering Systems Division, and principal research scientist in the Sociotechnical Systems Research Center (SSRC). She is the director of MIT's Systems Engineering Advancement Initiative (SEAri), a research group focused on advancing the theories, methods, and effective practice of systems engineering applied to complex sociotechnical systems. Prior to joining MIT in 2003, Dr. Rhodes held senior management positions in systems engineering and enterprise practices at IBM Federal Systems, Lockheed Martin, and Lucent Technologies.

Dr. Rhodes conducts research on innovative approaches and methods for architecting and design of complex systems and enterprises, including predictive indicators of performance, empirical studies of engineering systems thinking and practice, and designing for uncertain futures. Her research is driven by the desire to more predicatively architect socio-technical systems to address significant societal needs in a dynamic world. She is involved in research across multiple sectors including defense, aerospace, transportation, energy and commercial products.