Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 16 (2013) 532 – 541

Procedia Computer Science

Conference on Systems Engineering Research (CSER'13)
Eds.: C.J.J. Paredis, C. Bishop, D. Bodner, Georgia Institute of Technology, Atlanta, GA, March 19-22, 2013.

Exploiting stand-in redundancy to improve resilience in a system-of-systems (SoS)

Payuna Uday[a]*, Karen Marais[a]

[a]*Purdue University, 401 W. Stadium Avenue, West Lafayette, IN-47906, USA*

Abstract

Resilience is the ability of a system or organization to react to and recover from disturbances with minimal effect on its dynamic stability. While the resilience of system-of systems (SoSs) depends on the reliability of their constituent systems, traditional reliability approaches cannot adequately quantify their resilience. Given the heterogeneity and often wide geographic distribution of SoS constituent systems, inclusion of backup redundant systems for a SoS is usually impractical and costly. In this paper, we quantitatively assess the impact of compensating for a loss of performance in one constituent system by re-tasking the remaining systems. We call this "stand-in redundancy", and we develop two concepts to implement stand-in redundancy in a SoS. First, reactive resilience deals with performance recovery after a system failure has occurred. We provide a method to determine feasible alternative SoS configurations based on performance level recovery and cost of implementation. Second, proactive resilience takes into account the gradual degradation of systems over time. The corresponding reduction in SoS performance could initiate a forcible transition to a different SoS configuration before actual failure of the system. These concepts, and their resulting upstream effects on development costs and risks, can be used by decision-makers to quantitatively assess the impact on resilience of different SoS architectures and their inherent ability to resist failures throughout the SoS lifecycle.

Keywords: System-of-systems; resilience; stand-in redundancy; functional reconfigurability

1. Introduction

The emergence of complex systems over the past few decades has led to increased interest in exploring methods to incorporate high levels of inherent resilience within them. A complex system can be defined as " an open system with continually cooperating and competing elements – a system that continually evolves and changes according to its own condition and external environment" [1]. Examples of complex systems include satellites, aircraft, and the space shuttle. These systems are expensive to design and build, they operate in harsh or remote environments, and

* Corresponding author. Tel.: 1-765-464-4966; fax: +1-765-494-0307.
E-mail address: puday@purdue.edu.

doi:10.1016/j.procs.2013.01.056

any failure of these systems is typically a high publicity event. In some cases, such as satellites, maintenance and repair is difficult or impossible in physically inaccessible environments.

In recent years, networks of complex systems, known as system-of-systems (SoS), have garnered increased attention [2,3]. Specifically, the term *system-of-systems* is used to denote networks that are formed from the integration of independently operating complex systems that interact with one another to provide an overall capability, which cannot be achieved by the individual systems alone [1]. Examples of SoSs include the national air space (NAS) and the military's ballistic missile defense system. These meta-systems are characterized by the operational and managerial independence of the constituent systems, the evolutionary nature and emergent behavior of the larger SoS, and the geographic distribution of the sub-systems [4]. High levels of interdependency add to the overall complexity of the SoS. As a result, designing and operating a SoS is challenging both from an engineering as well as a managerial perspective. In particular, resilience in an SoS, though as vital as in the case of complex systems, is hard to capture and design through traditional means.

Resilience is the ability of a system or organization to react to and recover from disturbances at an early stage with minimal effect on its dynamic stability [5]. Typically, in large complex systems, redundancy features are used to increase the resilience of the system to perturbations. For instance, commercial satellites are fitted with multiple backup systems to limit performance loss in the event of failures. While resilience and redundancy are sometimes thought of as analogous, they are significantly different concepts. Redundancy is essentially the inclusion of secondary components or systems to provide operability when a primary system fails. It can, therefore, be thought of as an input to the design process, which ultimately provides some level of resilience (output) to the overall system. In other words, redundancy is just one way to achieve resilience in a system.

In this paper, we show how the inherent structure and traits of an SoS can be leveraged to improve the resilience of the overall network. We develop a method that combines the ability of the constituent systems to function independently with the evolutionary nature of the SoS to maintain dynamic stability in the event of a failure. That is, if one system fails, we compensate for this loss by re-tasking all or a subset of the remaining systems. We call this approach "stand-in redundancy", and propose two perspectives to address it:

1) Reactive resilience: Once a failure has occurred, we study different feasible configurations of the remaining systems that can be set up to meet the immediate needs of the SoS.
2) Proactive resilience: Given gradual performance degradation over time, we investigate the benefits of forcibly transitioning the SoS to a new configuration before actual failure of the system.

The remainder of this paper is organized as follows: Section 2 highlights the motivation behind this study, Section 3 describes the method developed, and Section 4 discusses the results of the analysis using an illustrative example. Section 5 concludes the paper.

2. Motivation

Traditional systems engineering practices try to anticipate and resist disruptions through classical reliability methods, such as inclusion of redundancy at the component level and use of preventive maintenance at the system level. Reliability analysis techniques, such as fault trees and event trees, are used to determine the level and types of redundancy to be included in the system design. Similar methods are used to develop maintenance plans to reduce the likelihood of failures at the system level.

However, these approaches do not adequately satisfy the resilience needs of a SoS. Given the heterogeneity and, often wide geographic distribution, of the constituent systems, redundant systems in a SoS are impractical and costly. Additionally, high levels of interdependency between the systems imply increased risks of failures cascading throughout the SoS. These hurdles offer the opportunity to improve the resilience of the overarching system through unconventional means. This view echoes that of researchers who raise the need for a different perspective of resilience in the context of SoSs [6,7]. A recent paper succinctly sums up the need for greater emphasis on resilient systems by stating that "systems should be made resilient, rather than merely reliable" as they "need to be able to recover from unexpected perturbations, disruptions, and degradations of the operational environment" [8].

Here, we study a way to compensate for a loss of performance in one constituent system by re-tasking the remaining systems. Specifically, as one entity, or node in a SoS, experiences degraded performance or a failure mode, other entities can alter their operations to compensate for this loss. We call this "stand-in redundancy". This concept raises several interesting questions, such as: (1) given the failure of a specific system, what is the best

configuration to compensate for the loss?; (2) what level of performance can be recovered with the new configuration?; and (3) what is the upstream effect of stand-in redundancy on development costs and risks?

To answer these questions, we develop two concepts: (1) reactive resilience, and (2) proactive resilience. *Reactive resilience* deals with performance recovery after a failure has occurred. In this case, for a specific capability, we study the reduction in overall SoS performance given various nodal failures, and then determine the level of performance that can be recovered by reconfiguring the rest of the SoS.

Having studied the impact of total nodal failures on overall SoS performance, we expand the method to track the impact of gradual degradation of nodes on the same overall performance. As the nodes degrade over time, the corresponding reduction in SoS performance could result in a situation where a different configuration might perform better than the current one. This implies that one might proactively transition the SoS to a different configuration before actual failure of the node. We call this *proactive resilience*, as this transition to a new configuration before nodal failure occurs improves the robustness of the overall SoS.

Figure 1 summarizes the above discussion and illustrates the different ways resilience in an SoS can be achieved. Failures can be addressed after they occur, through repair; or they can be anticipated and addressed before they occur, through preventive maintenance. Typically, exogenous methods are required to provide these services. Here, we emphasize designing systems with the inherent ability to react to failures, without causing any downtime for either maintenance and/or repair (as highlighted by the shaded region of the figure). This internal ability to resist failures can be achieved through means endogenous to the SoS. This alternative to traditional reliability engineering practices helps design systems-of-systems with inherent resilience by taking advantage of fundamental properties such as diversity, adaptability, and evolutionary behavior. These different kinds of resilience can be used to study various SoS architectures and evaluate their inherent ability to resist failures, thus in effect providing information for decision-makers to help identify "optimally" resilient SoS structures. Additionally, consideration of these resilience improvement techniques enables designers to better target risk resolution resources.

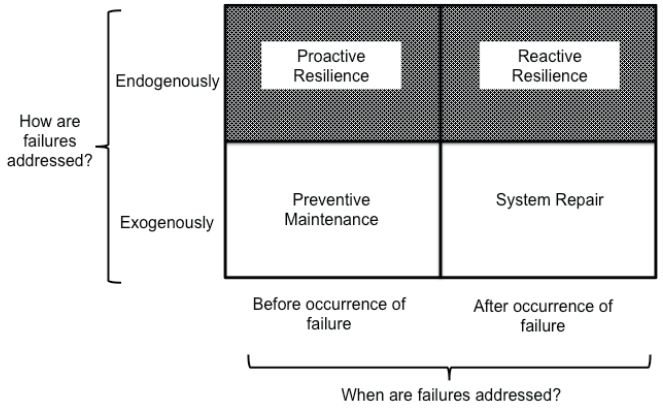

Figure 1. Classification of SoS resilience based on when and how failures are addressed

3. Analytic framework

3.1. Representation of an SoS

We consider a system-of-systems that consists of *n* systems. Typical representations of a SoS involve networked combinations of the constituent systems that ultimately provide SoS-level capabilities, as shown in Figure 2a. In this work, we represent a SoS as a hierarchical structure of multi-system capabilities and single-system functions, as shown in Figure 2b. At the highest level, the SoS is essentially a collection of the capabilities that it is designed to provide. Subsequently, each capability emerges from a collection of system-level functions.

At the lowest level, the functions are performed by the individual systems. For example, geosynchronous satellites can *image* large swathes of area for extended periods of time, and weapon-fitted UAVs can be used to

strike targets in hostile environments. Consideration of these functions rather than the individual systems as the base level of SoS capabilities allows us to analyze the resilience of the larger system. As mentioned previously, the traditional practice of incorporating redundancy in complex engineered systems may not fit well with respect to evolving SoSs. Thus, this view of recapturing lost or deteriorating functions plays a key role in improving the resilience of SoSs. In the above example, if surveillance is a key requirement of the mission and if the satellite fails, then with appropriate imaging capabilities, the UAV can be re-tasked to perform surveillance functions and to network with systems that were originally linked to the satellite.

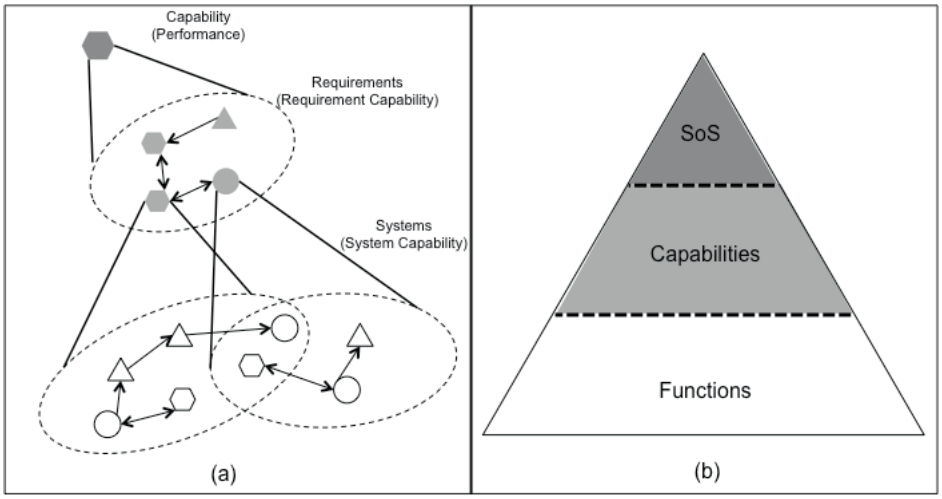

Figure 2. (a) Physical [9] and (b) functional representation of system-of-systems

At the middle level, groups of functions work together to provide a higher-level capability (C). For example, the satellite could collaborate with a reconnaissance UAV to not only provide surveillance of a large area but also to provide high-definition imaging capabilities for target identification. Performance and reliability are two important metrics with respect to any operational system. In this work, we consider these measures at the capability level and refer to them as the Level of Performance (LoP) and Level of Reliability (LoR) respectively.

Determination of the level of performance of SoS capabilities is challenging when compared to calculating these metrics for simpler systems. The level of performance achieved by a particular system function can be expressed by a direct measure of how well it performs its task. For instance, a direct measure of performance might be the area, say 1000 sq. mi., that a satellite can image with a given level of resolution. In contrast, determination of the level of performance for a particular capability is context and SoS dependent. As shown in Equation (1), computing this metric relies on a number of factors such as the architecture of the constituent systems, the availability of these systems, the performance capabilities of each system, and the functions achievable by each system.

$$LoP(C_i) = \mathcal{F}(constituent\ systems, functions, performance\ metrics, interdependencies) \quad (1)$$

The level of reliability provided by a capability is relatively simpler to determine. There are well-established methods that help determine the reliability of systems [10]. Using these traditional methods and tabulated failure rate information, the reliability of a system ($R_s(\mathrm{t})$), and by extension the reliability of the function it provides, at any time after it has been deployed can be computed. For this work, we define the level of reliability (LoR) of a capability as the probability that all the *nc* constituent systems contributing to that particular capability are operational at a particular time.

$$LoR(C_i) = Pr(all\ constituent\ systems\ are\ operational\ at\ time\ t) = \prod_{j=1}^{nc} R_{sj}(t) \quad (2)$$

3.2. Framework

Figure 3 provides a graphical illustration of SoS operations using the metrics of Level of Performance (LoP) and Level of Reliability (LoR) described above. The desirable region of operation is in the top right of the graph, that is, the high-performance, high-reliability portion (as indicated by the blue circle). After the initial deployment, the systems gradually degrade with time, and the SoS region of operation moves to the left of the graph. In some cases, this degradation can result in a simultaneous reduction in performance levels as well as reliability (not shown in figure). If a system fails, the immediate loss in its functionality leads to a decrease in the overall performance level of the SoS (as shown by the red circle). Given the inherent characteristics of SoSs, a single system loss typically does not lead to a complete failure of the larger SoS and hence, the overall LoP does not fall to zero. However, we propose that incorporating a certain level of stand-in redundancy will allow the SoS to minimize this performance loss without relying on external agents to either maintain or replace the failed system. This idea is illustrated in Figure 4. In the event of a system failure, by allowing multiple systems in the SoS to perform the same functions, the remaining systems in the SoS can be re-tasked to perform the lost functions, even if to a lesser degree of performance. This is represented by a sequence of green circles, indicating that different levels of performance can be regained depending on the functional reconfigurability of the SoS.

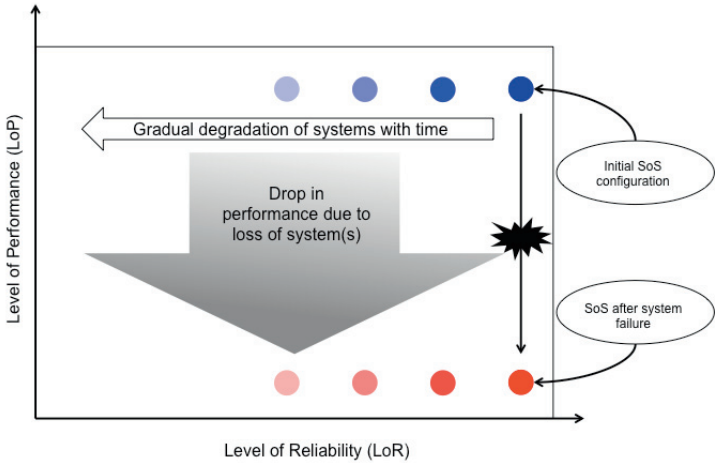

Figure 3. Notional LoP-LoR graph showing impact of system failure on SoS performance

The framework to improve SoS resilience using stand-in redundancy is based on a combinatorial optimization approach. This method aims to choose the SoS configuration that minimizes the operational cost of the SoS, while achieving a certain level of capability performance, as well as a threshold level of reliability. For a SoS with m capabilities, the optimization formulation thus becomes:

Minimize:

$$SoS\ operations\ cost \tag{3}$$

Subject to:

$$LoP_i \geq LoP_i^T \tag{4}$$

$$LoR_i \geq LoR_i^T \tag{5}$$

3.2.1. SoS operations cost

The SoS cost depends on a variety of factors and is highly context dependent. For this work, we divide SoS operations into three broad situations:

E Fully functional state: The SoS is in its original configuration with no system failures. The SoS cost depends on

the number of systems contributing to the overall SoS capabilities as well as the cost of operating each system.

- System loss state: The SoS has suffered either single or multiple system failures and is now operating at a much lower level of performance. Here, the SoS cost depends on the operating costs of the remaining systems as well as costs that may be incurred in repairing and/or replacing the failed system.
- Re-tasked state: The remaining systems are re-tasked to recover some of the lost functionality after a system failure. In this situation, the SoS cost depends on the operating costs of the remaining systems, the acquisition costs to include additional features that enable this functional redundancy, and the marginal costs to re-configure the SoS so that systems can take on their new tasks.

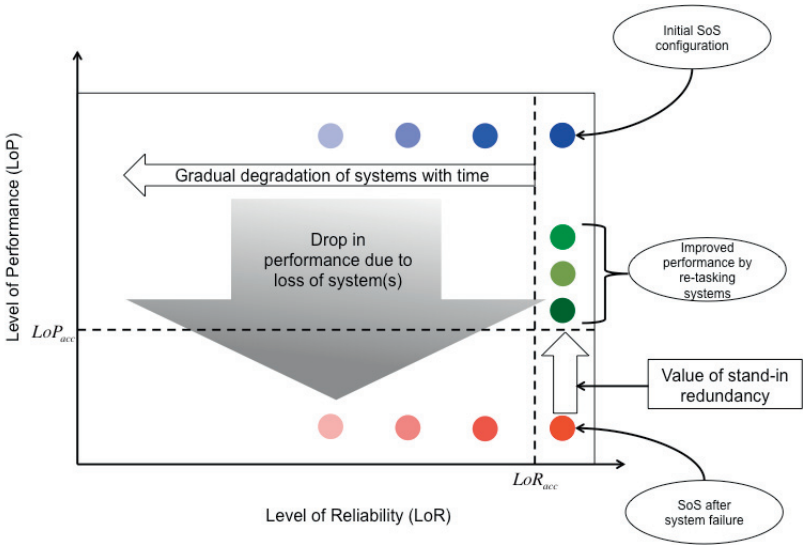

Figure 4. Notional LoP-LoR graph showing impact of functional reconfigurability on SoS performance

3.2.2. Level of Performance (LoP_i) and Level of Reliability (LoR_i)

These metrics are defined for each capability as described in Section 3.1.

3.2.3. Target values for Level of Performance (LoP_i^T) and Level of Reliability (LoR_i^T)

The constraints force the optimal solution to meet the target values of performance and reliability for each capability, defined by the decision-maker. In this study, these target values are defined under two broad categories:

- Desired target values (LoP_i^{des}, LoR_i^{des}): These values are used to determine the region of operation (in the *LoP-LoR* graph) of the original, fully functional SoS. They represent the level of performance and reliability the SoS is expected to satisfy. For example, for a fully functional and newly deployed SoS, the systems can be chosen to provide relatively high target values for each capability.
- Acceptable target values (LoP_i^{acc}, LoR_i^{acc}): These values are used to determine the region of operation (in the *LoP-LoR* graph) of the re-tasked SoS. They represent the minimum acceptable level of performance that the remaining systems must provide. As these values of LoP_i^{acc} and LoR_i^{acc} are varied, the costs to achieve the corresponding value of stand-in redundancy, vary accordingly.

4. Illustrative example

Consider the need for a SoS that provides the ability to detect and eliminate targets in hostile environments as well as provides large-scale surveillance in both hostile and non-hostile situations. The SoS consists of five systems: a geosynchronous satellite, three UAVs, and a ground station (see Figure 5). UAV-1 is primarily a surveillance drone with no weapons on board. On the other hand, UAV-2 and UAV-3 are fitted with weapons for target elimination tasks and they carry basic cameras on board to provide confirmation of the strikes that have been carried out. Combinations of these system functions yield three primary SoS capabilities: (1) surveillance, (2) target

identification, and (3) target elimination. Using the representation developed in Section 3.1, we decompose the SoS-level need into capabilities as shown in Figure 5. For example, to provide surveillance capabilities, the corresponding systems need to be able to image large areas of land with high revisit rates. Figure 6 indicates the functions that each system can provide. With this setup, the systems that contribute to the SoS capabilities are:

1. Surveillance provided by the satellite
2. Target identification provided by a collaboration between the satellite and UAV-1
3. Target elimination provided by UAV-2 and UAV-3

The main focus of this work is to assess the impact of incremental modifications to existing SoSs such that the overall architecture becomes more resilient to disruptions. With this focus in mind, we identify a few key ways this notional SoS can be modified. Due to challenges related to its accessibility, the features on the satellite cannot be changed. In contrast, it is relatively easier to retrofit the UAVs with higher performance devices, such as sophisticated imaging and communication equipment. Additionally, UAVs may also be reprogrammed for higher revisit rates. The grey arrows in Figure 6 indicate the systems whose corresponding features can be changed.

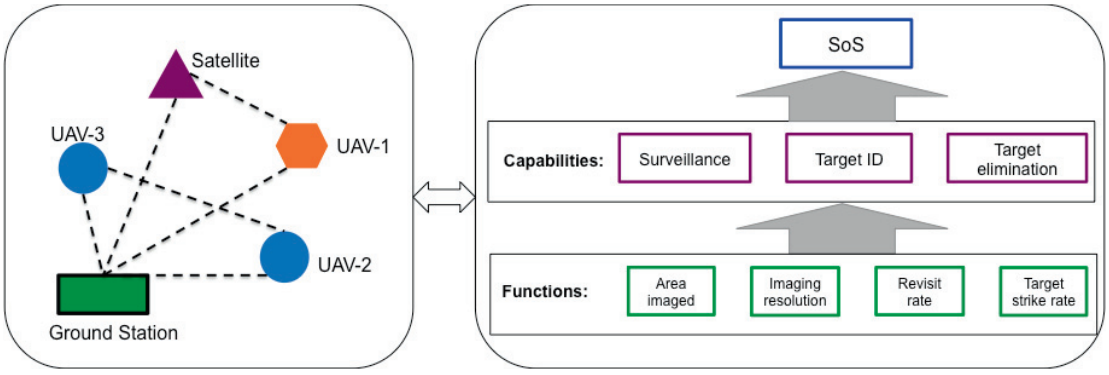

Figure 5. Five-system SoS and the corresponding decomposition of capabilities

Next, we determine the Level of Performance (LoP) and Level of Reliability (LoR) associated with each of the three capabilities. Both these metrics depend on the needs of the particular SoS, the systems available to perform tasks, and the specific functions that can be provided by these systems. In this study, we assess the LoP for each capability as the probability of achieving the particular capability. We know that combinations of system-level functions result in the corresponding capability. Let S_{fn} denote the set of system functions that contribute to a capability (C_i), and let S denote all possible combinations of these functions. Assuming binomial states of the systems, that is, each system is either fully functional or completely degraded (failed), we can determine the probability of achieving a particular capability by applying the law of total probability:

$$LoP(C_i) = \sum_S pr\left(C_i = 1 \middle| S_{fn} = S\right) pr(S_{fn} = S) \qquad (6)$$

Equation (6) means that the probability of achieving a certain capability depends on (a) the conditional probability that the capability is achievable given the operational status of the systems and the functions they provide, as well as (b) the probability that the systems, and by extension, their functions, are operational. The latter probabilities can be determined based on the reliability of the systems at the time of interest, as below. On the other hand, calculation of the conditional probability of success is relatively more complex since it depends on a variety of factors, such as the actual number of operational systems, the strength of the links between the systems, and the availability of sub-systems within the systems. The reliabilities of each of these systems can be calculated by applying historical failure rate information to Equation (2). We now perform the optimization to determine the least costly configuration under each of the following scenarios:

1. When nothing fails, that is, the original SoS performs as designed.
2. When one system fails.
3. When one or more systems are used to recover functionality after a system has failed.

System (x_i)	Functions (fn_i)			
	Area imaged (fn_1)	Imaging resolution (fn_2)	Revisit rate (fn_3)	Target strike rate (fn_4)
Satellite (x_1)	✔	✔	✔	-
UAV-1 (x_2)	✔ ↑	✔ ↑	✔ ↑	-
UAV-2 (x_3)	✔ ↑	✔ ↑	✔ ↑	✔
UAV-3 (x_4)	✔ ↑	✔ ↑	✔ ↑	✔

✔ Indicates feature present in system in the original SoS
↑ Indicates feature that can be upgraded/modified in system

Figure 6. Systems and the functions they provide

The results are shown in Figure 7. The horizontal axis denotes the system that has failed; the vertical axis represents the LoP for the capability under interest. For each failed system, the first bar indicates the original fully functional SoS (the baseline case for further comparisons), the second bar represents the impact of the system failure on a particular capability, and the last bar shows the effect of re-tasking the remaining systems in the SoS to achieve a minimum acceptable level of performance. For example, consider C_1 (surveillance) in Figure 7a. For this capability, the satellite alone provides the maximum LoP (100%) in the original SoS. If, however, the satellite fails, UAV-1 (with high definition imaging capabilities) can provide some surveillance capability (55%). On the other hand, if the systems were retrofitted with better imaging devices and reprogrammed for increased revisit rates, a combination of UAV-1 and UAV-2 are used to "stand-in" for the satellite, providing a higher LoP (72.5%) for the same capability.

In the case of C_2 (target identification) (see Figure 7b) loss of either the satellite or UAV-1 has a significant impact on this capability. Additionally, given the reliance of C_3 (target elimination) on the ability to accurately track a target, it is vital that drastic performance decrements, especially in urgent hostile situations, do not occur. When the satellite fails, assuming improved imaging capabilities on UAV-2, it can collaborate with UAV-1 to provide a marginally higher LoP. On the other hand, if UAV-1 were to fail, providing better imaging equipment on board UAV-2 (despite its primary role as an attack drone), raises the LoP by a significant amount (from 56% to 76%).

The impact of stand-in redundancy on C_3 leads to some interesting observations (see Figure 7c). While this capability directly stems from the targeting capabilities of the attack drones, namely UAV-2 and UAV-3, it relies heavily on accurate target identification information provided by C_2. Intuitively, the loss of either of the attack UAVs results in performance losses. For example, failure of UAV-2 alone leads to a direct loss of half the weapons striking capability of the SoS, and this loss cannot be recovered by any other system in the SoS as none of them are equipped to carry and launch weapons. In contrast, failure of either the satellite and/or UAV-1 adversely impacts the target identification ability of the SoS, thereby hindering its target elimination ability, despite the full functionality of UAVs 2 and 3. This highlights the fact that system failures can have significant impacts on capabilities that they are not directly designed to satisfy. In such situations, allowing other systems to take over some of the lost functionality helps maintain key capabilities at acceptable (as determined by decision-makers or operators) levels. For example, if UAV-1 fails in the midst of a raid, a properly equipped UAV-2 can provide target identification capability so that UAV-3 can carry out the actual target elimination tasks. Although this implies that UAV-2 may not be available to carry its own attack function, depending on the criticality of the situation, this marginal loss in firepower may be traded in for improved reconnaissance. As a result, it is important to keep in mind the immediate needs of the mission when deciding which configuration to transition to.

Cost considerations are highlighted in Figure 7a. $\Delta Cost_{re-tasked}$ is the cost associated with re-tasking the remaining system and it depends on: (a) the features that are either modified or added to the existing systems (such as, inclusion of high performance cameras in UAV-1 and UAV-2), and (b) the operating costs of these systems. This metric is relatively easier to compute than the corresponding cost for the failed system, $\Delta Cost_{failed}$. Calculating this cost is challenging as it depends on: (a) the operating costs of the systems remaining in the SoS after a nodal failure, (b) the costs to replace the failed system (for example, deploy a new UAV to replace the failed one), as well as (c) the costs accrued in the downtime between system failure and system replacement.

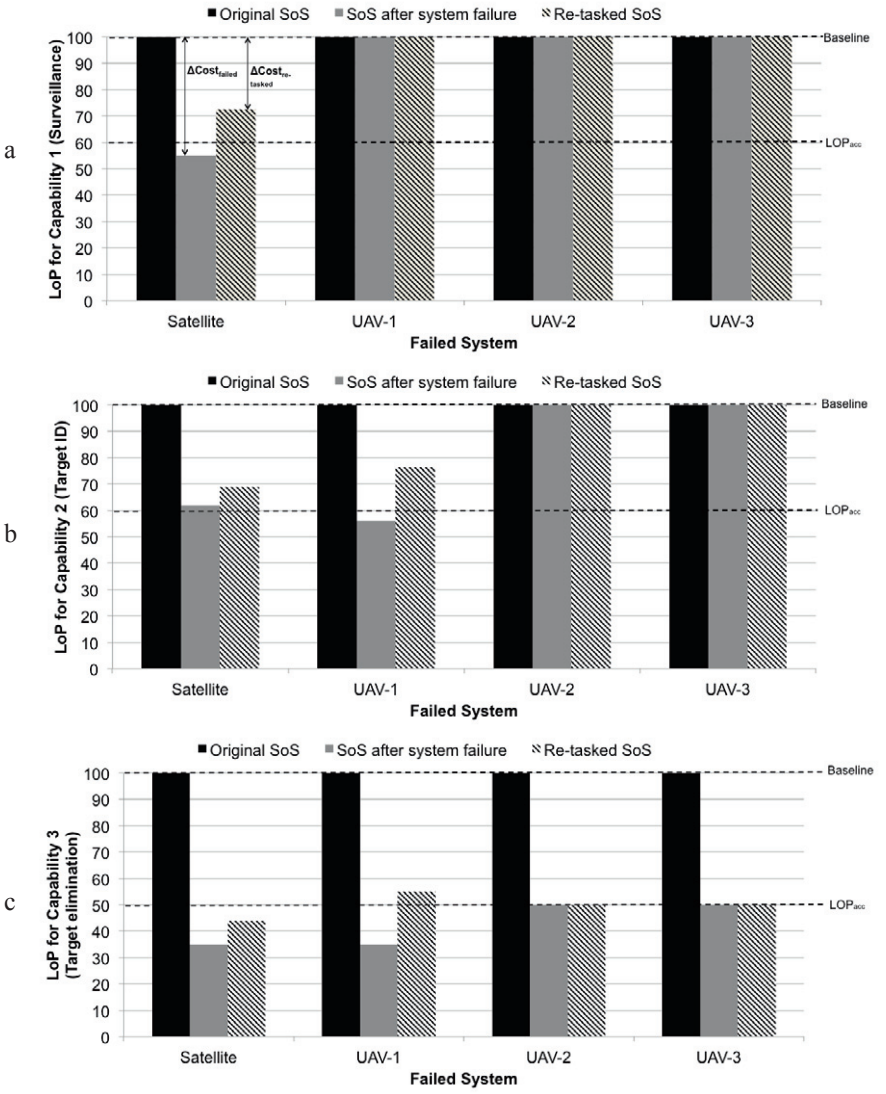

Figure 7. Impact of system failure and stand-in redundancy on (a) LoP of Capability 1 (Surveillance), (b) LoP of Capability 2 (Target identification), and (c) LoP of Capability 3 (Target elimination)

These results indicate that resilience in SoSs can be improved without having to use traditional practices of backing up systems. Instead, systems can be designed to contribute to SoS-level capabilities in the ideal case, and to "stand-in" for failed functions in the event of a failure. It is also important to note that there is a limit to the level of stand-in redundancy that can be incorporated in such systems. Returning to the notional example used in this study, re-tasking of some systems is possible because the UAVs can be retrofitted with better cameras and imaging mechanisms, and/or can be reprogrammed for higher revisit rates. On the other hand, UAVs used for purely surveillance purposes cannot be easily fitted with weapons deploying capabilities, nor can the designed imaging capabilities of the satellite be changed. This further leads to interesting implications regarding the balance between improving the resilience of an SoS, the costs associated with these improvements, and the need to incorporate features that not only improve the resilience but also ensure performance levels as the context of operations of the SoS changes with time. This is especially true of large-scale SoSs, such as multi-modal transportation networks, that are designed for long lifetimes with modifications and upgrades being incorporated in a gradual manner. Stand-in redundancy has the potential to improve the resilience of these systems, however, features and technological

modifications that bring about stand-in redundancy need to be chosen keeping in mind the trade-offs between costs, resilience in face of current and future threats, and adaptability of the SoS in an uncertain future.

5. Summary and future work

Traditionally, systems have been designed to be resilient through over-design. The emergence of large-scale system-of-systems (SoSs) has made it relatively hard to incorporate resilience in this manner as the system itself evolves with time along with its changing environment. Our approach indicates that incremental enhancements and/or modifications to existing systems in these SoSs can provide inherent resilience. The concept of stand-in redundancy, allows cost-effective re-tasking and reconfiguration of SoSs. The resulting resilience capability minimizes performance loss at the SoS level in the event of an unanticipated system failure, and thus, enables improved operability of the SoS through uncertain futures. The next step of this study includes expanding this static model to a dynamic one with the use of stochastic tools to design for resilience under uncertainty. Additionally, we will track the degradation of systems over time in order to assess whether it might be beneficial to force a transition to a different configuration before actual failure of a system (proactive resilience). While we limited the scope of this study to single system failures, ongoing research aims at investigating the implications of stand-in redundancy in the case of multi-system failures. These concepts, and their resulting upstream effects on development costs and risks, can be used by decision-makers to quantitatively assess the impact on resilience of different SoS architectures and their inherent ability to resist failures throughout the SoS lifecycle.

Acknowledgements

This material is based on work supported, in whole or in part, by the U.S. Department of Defense through the Systems Engineering Research Center (SERC) under Contract H98230-08-D-0171. SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology. The authors would like to thank Dr. William Crossley for insightful discussions that enhanced this work.

References

1. B. E. White, “Fostering Intra-Organizational Communication of Enterprise Systems Engineering Practices”, National Defense Industrial Association (NDIA), 9th Annual Systems Engineering Conference, San Diego CA, October 23-26, 2006.

2. D. DeLaurentis, W. Crossley, and M. Mane, “Taxonomy to Guide Systems-of-Systems Decision-Making in Air Transportation Problems”, Journal of Aircraft, Vol. 48, No. 3, pp 760-770, May-June 2011.

3. B.G. McCarter and B.E. White, "Emergence of SoS, Socio-Cognitive Aspects", Chapter 3 in M. Jamshidi's book, "System of Systems Engineering- Principles and Applications", 2007.

4. M. W. Maier, “Architecting Principles for System-of-systems”, Journal of Systems Engineering, Vol.1, No. 4, pp. 267-284, 1998.

5. E. Hollnagel, D. W. Woods, and N. Leveson, Resilience Engineering: Concepts and Precepts. Ashgate, 2010.

6. R. Neches and A. Madni, “Towards Affordably Adaptable and Effective Systems”, Journal of Systems Engineering, doi: 10.1002/sys.21234, October 2012

7. S. Sheard and A. Mostashari, “A Framework for System Resilience Discussions”, 18th Annual International Symposium of INCOSE, Utrecht, Netherlands, June 15-19, 2008.

8. A. Madni and S. Jackson, “Towards a Conceptual Framework for Resilience Engineering”, IEEE Systems Journal, Vol. 3, No. 2, pp. 181-191, 2009.

9. S. Y. Han, K. Marais, and D. DeLaurentis, “Evaluating System of Systems Resilience using Interdependency Analysis”, IEEE International Conference on Systems, Man, and Cybernetics, Seoul, Korea, October 14-17, 2012.

10. M. Rausand and A. Hoyland, System Reliability Theory: Models, Statistical Methods, and Applications. Second edition. New Jersey: Wiley – Interscience, 2004.