Framework for Managing System-of-Systems Ilities

Zhengyi Lian and Siow Hiang Teo

Abstract The design of ilities in System-of-Systems (SoS) architecture is a key means to manage changes and uncertainties over the long life cycle of an SoS. While there is broad consensus on the importance of ilities, there is generally a lack of agreement on what they mean and a lack of clarity on how they can be engineered. This article presents the DSTA Framework for Managing SoS Ilities, which coherently relates key ilities identified as important for SoS architectural design. Newly established in 2013 and updated in 2015 to guide Systems Architecting practitioners in DSTA, the framework also proposes how working definitions of robustness and resilience can be interpreted across key high-level and low-level ilities coherently, and introduces broad concepts of how they could be realized.

Keywords System-of-systems ilities · Robustness · Resilience · Evolvability · Flexibility

1 Introduction

A System-of-Systems (SoS) can be described as a set of constituent systems or elements that are operationally independent, but working together to provide capabilities that are greater than the sum of its parts. Managed independently and distributed geographically, these constituent systems work together to perform some high-level mission which cannot be accomplished by any individual constituent system alone [1]. Figure 1 shows an example of an SoS [2] in which a range

Z. Lian (✉) · S.H. Teo
Defence Science and Technology Agency (DSTA), 1 Depot Road, Singapore 109679, Singapore
e-mail: lzhengyi@dsta.gov.sg

S.H. Teo
e-mail: tsiowhia@dsta.gov.sg

M.-A. Cardin et al. (eds.), *Complex Systems Design & Management Asia*,
Advances in Intelligent Systems and Computing 426,
DOI 10.1007/978-3-319-29643-2_3

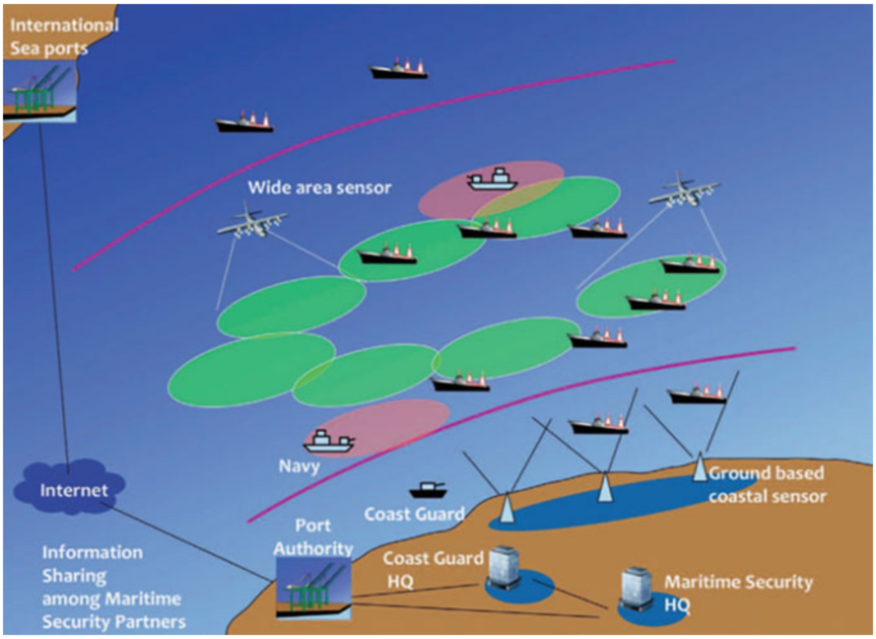


Fig. 1 Example of a maritime security SoS [2]

of systems like coastal and airborne sensors, vessels and information systems from various stakeholders are networked to perform a Maritime Security mission.

Ilities is a term used to describe desirable, lifecycle attributes (usually but not always ending in "ility") of SoS/systems that are not primary functional requirements but manifest themselves after the SoS/system had been put to initial use [3]. Some of such attributes are robustness, resilience, survivability, evolvability, flexibility, adaptability, interoperability, sustainability, reliability, availability, maintainability and safety. For example, robustness and resilience are especially important non-functional attributes for military systems as they are expected to continue performing under harsh environments (even when under attack) and to recover damaged constituents/elements quickly after an attack.

Robustness broadly means maintaining acceptable *performance* across variation in *context*. *Performance* can be interpreted at different levels, such as (a) high-level mission measure of effectiveness (MOE) in MITRE's concept of mission assurance [4], (b) lower level component system measure of performance (MOP) in the United States Department of Defence (U.S. DoD) concept of robust design [5]. Similarly, *context* can be interpreted broadly as mission, operating environment, associated operational contingencies (e.g. deliberate attack, internal failures), constraints, etc. Resilience, in accordance with INCOSE [6], encompasses the ability of a system to absorb (e.g. minimal or graceful degradation) the impact of a disruption or attack, stay at or recover to above an acceptable level of performance and sustain that level for an acceptable period of time.

2 Value and Hierarchy of Ilities

Ilities characterize a system's ability to respond to disturbances and changes in its environment, both foreseeable and unforeseeable. The design of ilities in SoS architecture is a key strategy to achieve consistency in value delivery and to manage changes and uncertainties over the long life cycle of an SoS. The goal is an enduring architecture that continues to perform well under various situations; yet remains flexible and forward-looking to evolve (e.g. allow insertion of new systems, adoption of new concepts of operation) according to changing threats, technology or stakeholder needs.

Despite the well-acknowledged value of ilities, there remains a lack of consensus on what they mean precisely and a lack of clarity of how they can be engineered, except in established areas like quality and RAMS (i.e. Reliability, Availability, Maintainability and Safety). A team comprising de Weck, Ross and Rhodes, from the Massachusetts Institute of Technology Engineering Systems Division (MIT ESD)—Systems Engineering Advancement Research Initiative, conducted a study [7] on 20 common ilities and their inter-relationships (based on co-occurrence in literature[1]), which suggests that a means-end hierarchy of ilities could exist, with certain ilities supporting other ilities. However, a universally accepted hierarchy has yet to be established.

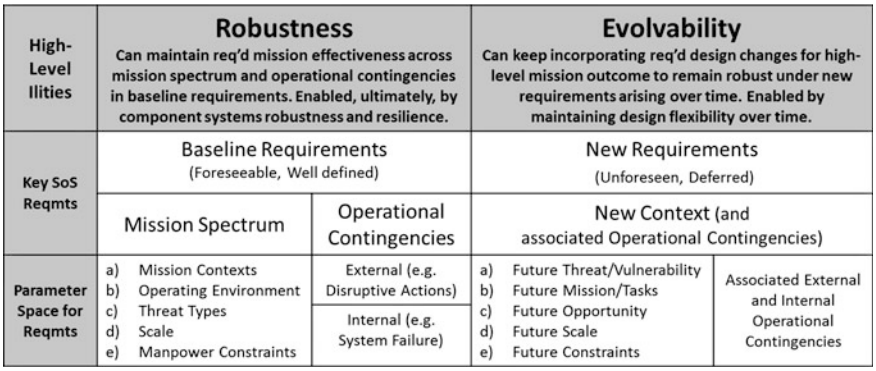
It is desired to create a practicable framework, in the context of the Singapore Armed Forces (SAF) and DSTA, which identifies the key ilities and their relationships to guide the design of ilities in systems architectures. This is the motivation behind the DSTA Framework for Managing SoS Ilities.

3 High-Level (SoS Key Mission) Ilities

We interpret the desired goal for SoS design as Robustness in fulfilling some key high-level mission (i.e. high-level Robustness) under all circumstances. However, practical constraints limit this goal to the mission spectrum and associated operational contingencies defined under a set of foreseeable, well-defined baseline requirements spanning multiple years (see Robustness column of Table 1).

[1]Scientific papers within the Inspec and Compendex database (1884–2010) were searched to rank the prevalence of the 20 ilities and found to be consistent with that based on Google search engine hits. Thereafter, an internet search of how often two ilities co-occurred in the same article/page was used to estimate the strength of their inter-relationship.

Table 1 High-level ilities: robustness (in key mission outcome) and evolvability [2]



High-Level Ilities	**Robustness** Can maintain req'd mission effectiveness across mission spectrum and operational contingencies in baseline requirements. Enabled, ultimately, by component systems robustness and resilience.		**Evolvability** Can keep incorporating req'd design changes for high-level mission outcome to remain robust under new requirements arising over time. Enabled by maintaining design flexibility over time.	
Key SoS Reqmts	Baseline Requirements (Foreseeable, Well defined)		New Requirements (Unforeseen, Deferred)	
	Mission Spectrum	Operational Contingencies	New Context (and associated Operational Contingencies)	
Parameter Space for Reqmts	a) Mission Contexts b) Operating Environment c) Threat Types d) Scale e) Manpower Constraints	External (e.g. Disruptive Actions)	a) Future Threat/Vulnerability b) Future Mission/Tasks c) Future Opportunity d) Future Scale e) Future Constraints	Associated External and Internal Operational Contingencies
		Internal (e.g. System Failure)		

Mission spectrum can be described in more detail by parameters such as context, area of operations (e.g. extent and nature), potential threat types, scale (e.g. quantity, level of coordination) and manpower (or other strategic resource) constraints. Operational contingencies can be classified as originating (i) externally, such as disruptive actions (e.g. jamming, cyber-attack, physical damage) by an adversary, extreme natural phenomena (e.g. lightning strikes, heavy storms, earthquakes), or (ii) internally, such as equipment failure, accidents, logistic demand spikes.

High-level Robustness can be measured as the percentage of this baseline requirements parameter space where the high-level mission measure of effectiveness (MOE) stays above some required threshold. In reality, this is challenging due to the many attributes and scenarios to be evaluated in the parameter space. A practical approach is to rationalize and categorize the scenarios by impact and likelihood, followed by further decision on what to include in baseline requirements for high-level Robustness evaluation and what to defer.

The planning space of baseline requirements for high-level Robustness spans multiple years (see Fig. 2). This multi-year planning space can change over the SoS life-cycle to accommodate context changes and new/revised requirements (unforeseen or consciously deferred at the design stage) arising in future.

For example, uncertainty over the materialization of a threat and/or a lack of cost-effective technological solutions may lead to the deferment of requirements to a later stage. These requirements may be re-incorporated when there is more clarity in the threat or solution space. The new requirements will merge with the original baseline requirements to form the newly evolved baseline requirements for the SoS. Some aspects of the original baseline requirements that have become irrelevant over time will be removed or revised in the process.

As another example, abrupt shifts in the Maritime Security landscape (e.g. rise of non-state actors engaging in piracy) may necessitate a significantly higher tempo of operations and/or more surveillance nodes to be established for a Maritime Security

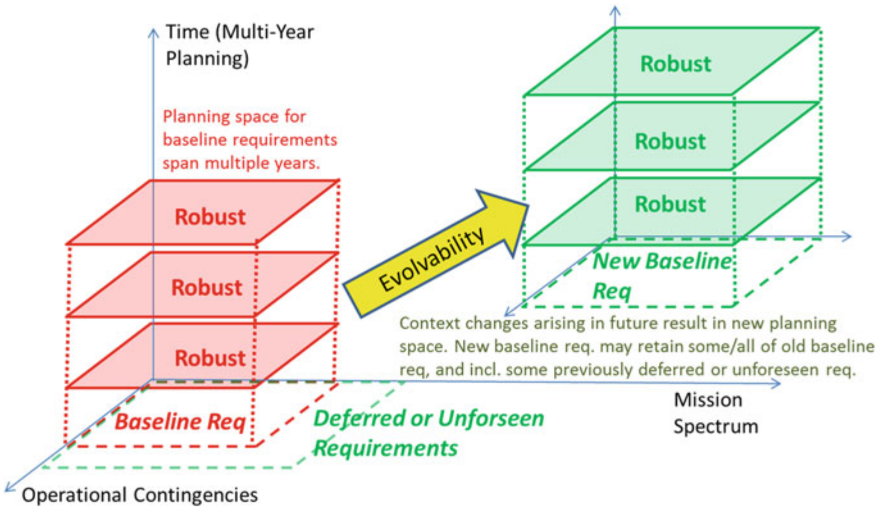

Time (Multi-Year Planning)
Planning space for baseline requirements span multiple years.
Robust
Robust
Robust
Baseline Req
Deferred or Unforseen Requirements
Operational Contingencies
Evolvability
Robust
Robust
Robust
New Baseline Req
Context changes arising in future result in new planning space. New baseline req. may retain some/all of old baseline req, and incl. some previously deferred or unforeseen req.
Mission Spectrum

Fig. 2 Planning for high-level robustness spans multiple years. Evolvability is key to maintaining that robustness over time as baseline requirements change

SoS. These changes may create permanent stress to the existing SoS infrastructure (e.g. communications architecture or logistics chain) in a way that necessitates changes to the SoS design. The availability of new technologies (e.g. commoditization of satellite imagery, maturation of fully autonomous patrol vessels) may also offer opportunities for force transformation. These possible shifts in context are likely to translate to revised requirements for the SoS.

Evolvability will allow greater ease to keep incorporating relevant design changes for the SoS to remain robust in fulfilling the high-level mission under each new planning space as new requirements arise over time (see Fig. 2 and Evolvability column of Table 1). This may be measured by the cumulative cost of transition, in terms of capital costs, engineering complexity and/or manpower, associated with anticipated design changes over time, for the purpose of relative comparison between SoS alternatives. In view of the long time horizon and uncertainty of future requirements, such measurements can only be limited to foreseeable new requirements over specified time frames (i.e. the known unknowns).

Hence, Robustness and Evolvability are the two key high-level ilities that an SoS architecture should possess to meet baseline requirements of some key high-level mission, while maintaining design flexibility to meet new/revised requirements (with design change implications) arising over time.

4 Lower-Level (Component Systems Mission or Performance) Ilities

The Robustness of an SoS in fulfilling its key high-level mission (i.e. high-level Robustness) depends on the robustness and resilience of its component systems in fulfilling their lower level missions, which can ultimately be traced to attaining required levels for a set of measures of performance (MOPs) (see Table 2 for examples). Those MOPs should ideally be measurable over time to monitor their levels during normal operations, as well as their degradation (extent and duration) in the event of an operational contingency or disturbance (see Fig. 3).

At the component systems level, robustness can be interpreted as attaining required levels for those MOPs across the SoS mission spectrum and maintaining them under associated operational contingencies (see graphic (A) in Fig. 4). The objective here is for a sufficient subset, *not all*, of the component systems to remain robust under each contingency, so that Robustness in the high level SoS key mission is preserved (see Fig. 5 for concept illustration). In order to sustain this across successive contingencies, component systems resilience is vital to rebuilding capacity to absorb damage from the next wave of contingencies, by limiting damage and recovering affected MOPs of component systems that are more severely impacted by and failed to remain robust under the current wave of contingencies (see graphic (B) in Fig. 4).

There may be cases of contingencies so severe or drastic in impact that existing resilience mechanisms can only recover affected component systems partially. Lessons learnt from such setbacks could trigger a redesign with the aim of recovering affected MOPs to normal/better levels and maintaining them above

Table 2 Examples of SoS/high-level mission and associated lower level component systems, missions and MOPs

SoS and high level mission	Air-power generation	Maritime security
High-level MOE	Probability and time taken to generate req'd level of air-power an attack	Probability and time taken to detect and respond to maritime incidents and threats
Component systems	Individual air-bases or APG clusters	Coastal and sensors, patrol vessels, information systems
Lower level missions	Air-craft generation air-base operability	Surveillance patrol and boarding operations
Possible MOps at a lower level can be related to	Number of aircrafts on standby Aircrafts preparation time Logistics resupply time L/R platform operability L/R platform inspection, recovery key assets and resource availability	Coverage (persistent) of coastal sensors Coverage (transient) revisit time of air-borne sensors and patrol vessels Time to reach incident/threat location Time for completion of boarding operations key assets and resource availability

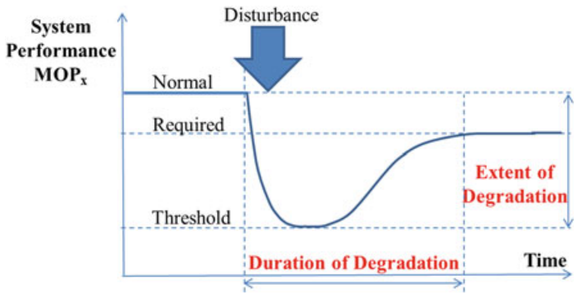


Fig. 3 Disturbances can degrade specific component system MOPs. It is desired for MOP to recover to normal (or at least required) levels quickly after the disturbance, but that may be impossible if the extent of degradation has exceeded some threshold (referenced and adapted from concepts on survivability from MIT ESD [8, 9] and resilience from the Future Resilient Systems project at the Singapore–IEH Centre [10])

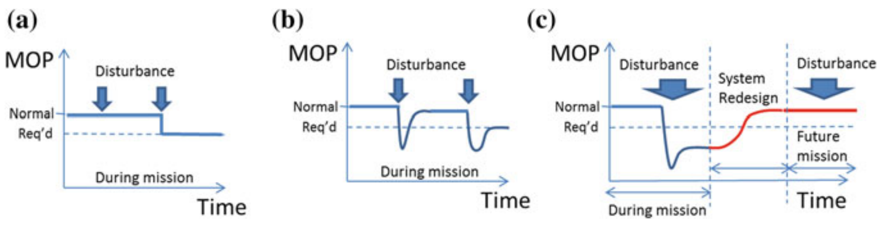


Fig. 4 Component systems responses to operational contingencies: **a** maintain MOP, **b** limit extent and duration of degradation to MOP, **c** design change

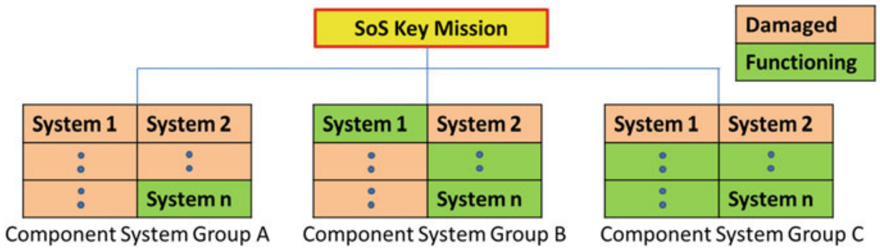


Fig. 5 High-level (SoS key mission) robustness can be preserved after an operational contingency as long as a sufficient subset of component systems (in *green*) remain robust (i.e. relevant MOPs can still attain levels required for mission success)

required levels should the same severe contingencies strike again in future (see graphic (C) in Fig. 4). The flexibility which this design change can be achieved (i.e. design flexibility) without compromising continuity of current operations contributes to SoS evolvability.

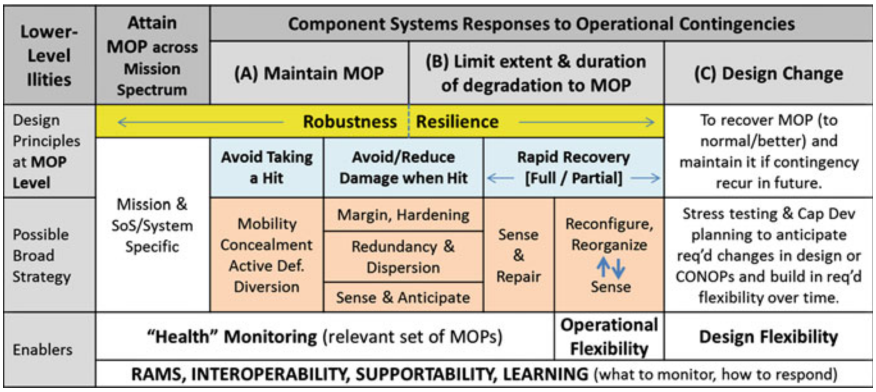
Ideally, it would be good to anticipate such severe contingencies and evaluate their impact, either through simulations or exercises, so that the necessary design changes/flexibility can be planned for and implemented progressively at the right opportunity. Possible design choices to preserve design flexibility over time may include (i) availability of redundant component systems to experiment with new designs while preserving capability to perform SoS key missions, (ii) modular task force design to allow easy replacement of platforms or mission modules for different tasks, (iii) avoiding tight coupling, as well as use of closed, proprietary data standards and communication protocols, among component systems.

Table 3 summarizes the above discussion on lower-level ilities from the perspective of component systems responses to operational contingencies. In short, component systems robustness (by attaining required levels for a set of MOPs across the mission spectrum and maintaining them under contingencies) and resilience (by limiting the extent and duration of degradation to those MOPs levels under contingencies) contributes to high-level (SoS key mission) Robustness. Flexibility for design change without compromising continuity in current operations contributes to SoS Evolvability.

5 Achieving Component Systems Robustness and Resilience

Table 3 also represents an attempt to distil out the key design principles, possible broad strategies and enablers that contribute to component systems robustness and resilience, as a guide to generation of solution alternatives for implementation.

Table 3 Component systems responses to operational contingencies: design principles, possible broad strategies and enablers



Lower-Level Ilities	Attain MOP across Mission Spectrum	Component Systems Responses to Operational Contingencies				
		(A) Maintain MOP		(B) Limit extent & duration of degradation to MOP		(C) Design Change
Design Principles at MOP Level	← Robustness	Resilience →				To recover MOP (to normal/better) and maintain it if contingency recur in future.
		Avoid Taking a Hit	Avoid/Reduce Damage when Hit	Rapid Recovery ← [Full / Partial] →		
Possible Broad Strategy	Mission & SoS/System Specific	Mobility Concealment Active Def. Diversion	Margin, Hardening Redundancy & Dispersion Sense & Anticipate	Sense & Repair	Reconfigure, Reorganize ↑↓ Sense	Stress testing & Cap Dev planning to anticipate req'd changes in design or CONOPs and build in req'd flexibility over time.
Enablers	"Health" Monitoring (relevant set of MOPs)				Operational Flexibility	Design Flexibility
	RAMS, INTEROPERABILITY, SUPPORTABILITY, LEARNING (what to monitor, how to respond)					

The attainment of required levels for a set of MOPs across a baseline mission spectrum under normal conditions, usually via some mission and SoS/system specific design features, is a typical pre-requisite to be verified during system acceptance tests. However, the above is insufficient to assure component systems robustness and resilience under operational contingencies. That would require additional design features to respond to contingencies along the principles of (1) avoid taking a "hit", (2) avoid/reduce damage when "hit", and (3) rapid recovery.[2] As illustrated in Table 3, principles (1) and (3) contribute to component systems robustness and resilience respectively, while principle (2) contributes to both.[3] These are supported by the enablers "health" monitoring and operational flexibility as shown in Table 3. "Health" monitoring here refers to sensing systems to (i) verify relevant MOPs can attain and maintain required levels over time, (ii) monitor extent of MOP degradation during contingencies for deciding when and where to activate recovery mechanisms, (iii) quantify time taken for recovery. Operational flexibility here encompasses the (i) agility of commanders and operators to quickly respond to contingencies in unfamiliar or untested situations, and (ii) their adaptability to modify responses while sensing overall effectiveness in recovering from the contingency.

5.1 *Principle (1)—Avoid Taking a "Hit"*

To avoid relevant MOPs being degraded by a disturbance from an external/internal source (i.e. "hit"), possible broad strategies include:

(a) **Mobility**. To stay away from the area of influence of that disturbance.
(b) **Concealment**. To not be easily distinguishable from the environment and be targeted by the source of that disturbance.
(c) **Active Defence**. An active layer to eliminate that disturbance itself or its source, or at least prevent the disturbance from penetrating the boundaries of a system being targeted.
(d) **Diversion**. To deflect/redirect a disturbance that has penetrated a system's boundaries away from critical components of the system.

[2]The three principles here are an explicit description of the concepts underlying MIT ESD's survivability design principles of (1) reduce susceptibility, (2) reduce vulnerability and (3) enhance resilience [8, 9].

[3]At the implementation level, we see robustness and resilience as complimentary and overlapping. Robustness measures (avoid a hit, reduce damage when hit) can maintain MOP above required levels for disturbances up to a certain severity. Beyond that, resilience takes over, whereby the same measures to "reduce damage when hit" under robustness now acts to limit damage so that recovery is possible, followed by quick recovery within the mission time-frame. Resilience operates at the tactical level in response to imminent disturbances; evolvability operates at the strategic level in preparation for new/deferred requirements.

5.2 *Principle (2)—Avoid/Reducing Damage When "Hit"*

When the above layer fails to avoid a component system being "hit" by a disturbance, possible broad strategies to avoid or limit the damage caused include:

(a) **Margin, Hardening**. Building in sufficient margin between required and normal MOP levels allows for some degradation in MOP before the high-level mission effectiveness is compromised. This can be complemented with "hardening" of critical components to resist or slow down the rate of MOP degradation, which can help to lower the amount of margin required.

(b) **Redundancy and Dispersion**. This refers to duplication and avoiding co-location of critical assets/capabilities/operations to avoid creating single-point vulnerabilities/failures. If dispersion via geographical separation is impractical, containment through appropriate barriers to reduce failure propagation (e.g. network separation for IT systems) can be an alternative strategy.

(c) **Sensing and Anticipation**. This applies more to the people or organizations operating an SoS, making sense of anomalies and warning indicators picked up through existing intelligence, sensors or monitoring systems to anticipate an imminent disturbance or detect a developing situation before the damage done becomes too serious. The advance warning can allow operators to respond appropriately (e.g. take cover and avoid exposure, activate barriers or further hardening, getting ready for activation of contingency plans) to reduce potential damage to a minimum. However, a challenge to such a strategy is false alarm and the associated costs. Hence, a fundamental enabler for such a resilience capability is organization learning from past and on-going experience on what to sense/monitor as well as how to respond appropriately to each early warning indicator. (Adapted from Erik Hollnagel's [11] resilient systems/organizations of the third kind.[4])

5.3 *Principle (3)—Rapid Recovery*

When damage is inevitable after a disturbance, the presence of recovery mechanisms is important to restore affected MOPs quickly and rebuild the capacity to withstand subsequent disturbances. Two possible categories of strategies are:

(a) **Sense and Repair**. "Sense" here refers to using "health" monitoring systems to detect degradation in relevant MOPs, followed by root-cause identification and damage location. Next, decision support tools can help in directing appropriate repair resources to the damage location in the most efficient manner. If the

[4]Resilient systems/organizations of the third kind can anticipate and manage something before it happens, instead of passively responding to something that happens.

severity of damage can be limited, this category of response can typically achieve a full recovery to affected MOPs. (Adapted from Erik Hollnagel's [11] resilient systems of the first kind.[5])

(b) **Reconfigure/Reorganize $\longleftrightarrow$ Sense**. When damage from a disturbance is so severe that only a partial recovery is possible with existing repair resources, the next strategy is to reconfigure whatever functional capabilities/assets remaining, or to reorganize existing resources, to focus on priority tasks for fulfilling the high-level mission. Contingency planning can help to achieve a quick first response here. However, as the circumstances associated with each contingency evolves differently, a key enabler to deal with such complexity is the operational flexibility of commanders and operators on the ground to regularly "sense" how well the reconfiguration/reorganization is performing and adapt through iterative adjustments to arrive at something that works. Passing down lessons learnt from such episodes can help to sharpen this resilience capability through either better contingency planning for a better first response, or knowing what signals to monitor in making timely adjustments. (Adapted from Erik Hollnagel's [11] resilient systems of the second kind.[6])

6 Fundamental Enablers

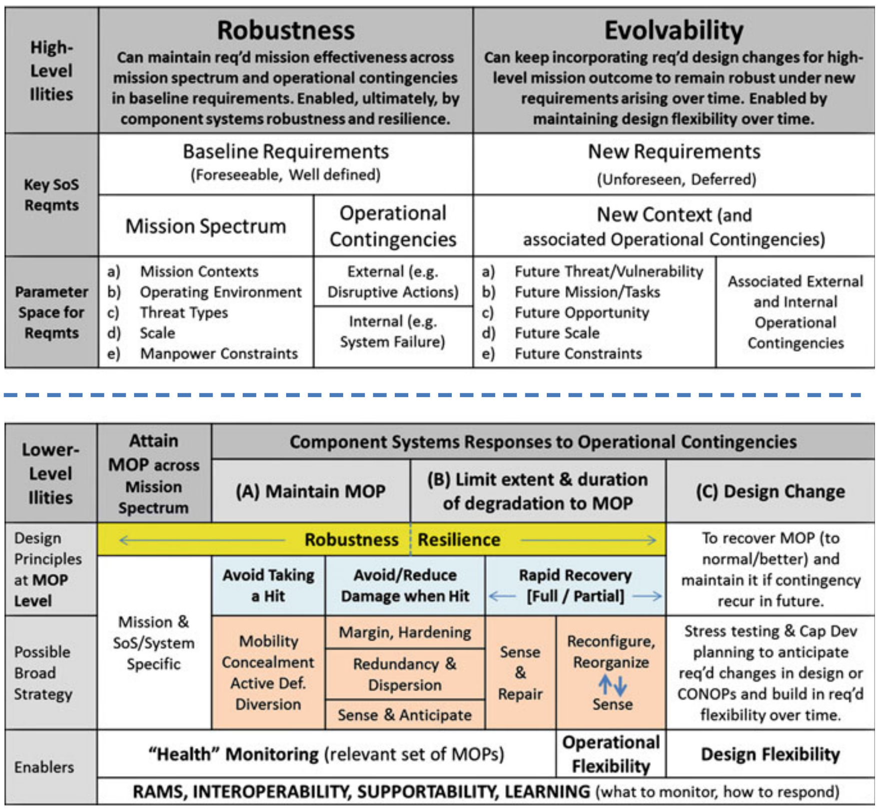
The following enablers are fundamental to sustaining effective operations of an SoS over its long life-cycle, which is a pre-requisite to achieving high-level Robustness under baseline (current) and new (future) requirements.

(a) **RAMS**. To operate in an effective and safe manner continuously during the mission period, as well as to instil confidence in sustained mission-readiness over the SoS life-cycle.
(b) **Interoperability**. To enable constituent systems or elements to provide and/or accept data, information, material and services from one another to work coherently in achieving a desired operational effect or high-level mission.
(c) **Supportability**. To sustain effective operations for component systems robustness and resilience over the SoS life-cycle as efficiently as possible, in light of specific resource (e.g. manpower) constraints, through appropriate SoS design (e.g. workflow, technology) at the front end or subsequent SoS evolution.
(d) **Organization Learning**. To accumulate and pass down experience on (i) what to monitor to verify component systems robustness, anticipate potential contingencies, assess need for and effectiveness of responses during contingencies;

[5]Resilient systems/organizations of the first kind can monitor situations to determine if a reaction is necessary, and thereafter respond appropriately.

[6]Resilient systems/organizations of the second kind can manage (i.e. monitor and respond to) something not only when it happens, but also learn from what has happened thereafter to adjust both what it monitors and how it responds.

Table 4 DSTA framework for managing SoS ilities



High-Level Ilities	**Robustness** Can maintain req'd mission effectiveness across mission spectrum and operational contingencies in baseline requirements. Enabled, ultimately, by component systems robustness and resilience.		**Evolvability** Can keep incorporating req'd design changes for high-level mission outcome to remain robust under new requirements arising over time. Enabled by maintaining design flexibility over time.	
Key SoS Reqmts	Baseline Requirements (Foreseeable, Well defined)		New Requirements (Unforeseen, Deferred)	
	Mission Spectrum	Operational Contingencies	New Context (and associated Operational Contingencies)	
Parameter Space for Reqmts	a) Mission Contexts b) Operating Environment c) Threat Types d) Scale e) Manpower Constraints	External (e.g. Disruptive Actions) Internal (e.g. System Failure)	a) Future Threat/Vulnerability b) Future Mission/Tasks c) Future Opportunity d) Future Scale e) Future Constraints	Associated External and Internal Operational Contingencies

Lower-Level Ilities	Attain MOP across Mission Spectrum	Component Systems Responses to Operational Contingencies				
		(A) Maintain MOP		(B) Limit extent & duration of degradation to MOP		(C) Design Change
Design Principles at MOP Level	← Robustness		Resilience →			To recover MOP (to normal/better) and maintain it if contingency recur in future.
	Mission & SoS/System Specific	Avoid Taking a Hit	Avoid/Reduce Damage when Hit	← Rapid Recovery [Full / Partial] →		
Possible Broad Strategy		Mobility Concealment Active Def. Diversion	Margin, Hardening Redundancy & Dispersion Sense & Anticipate	Sense & Repair	Reconfigure, Reorganize ↑↓ Sense	Stress testing & Cap Dev planning to anticipate req'd changes in design or CONOPs and build in req'd flexibility over time.
Enablers	"Health" Monitoring (relevant set of MOPs)				Operational Flexibility	Design Flexibility
	RAMS, INTEROPERABILITY, SUPPORTABILITY, LEARNING (what to monitor, how to respond)					

(ii) how to respond (repair, reconfigure, reorganize) and adapt to changing circumstances during contingencies. This will contribute to component systems resilience, and ultimately high-level Robustness.

7 Conclusion

The design of ilities in SoS architecture is a key means to manage changes and uncertainties in environment, technology and stakeholder needs over the long SoS life cycle. The DSTA Framework for Managing SoS Ilities (summarized in Table 4) pioneers the effort to identify and coherently relate key high-level and low-level ilities for an enduring SoS architectural design. In summary, Robustness and Evolvability to are the two key high-level ilities that an SoS architecture should possess. At a lower level, component systems robustness and resilience contribute to high-level Robustness; flexibility for design change without compromising continuity in existing missions contributes to Evolvability. Component systems robustness and resilience to disturbances can be enhanced through strategies along the principles of (1) avoid taking a "hit", (2) avoid/reduce damage when "hit" and (3) quick recovery, supported by SoS "health" monitoring and operational flexibility. Fundamental enablers to sustain effective operations over the SoS life-cycle include RAMS, interoperability, supportability and organization learning.

Acknowledgements The authors would like to thank Mr. Kang Shian Chin and Mr. Sim Kok Wah for initial development of this framework published in DSTA Horizons 2013/14 [2], as well as the perspectives and inputs of Mr. Kang Shian Chin, Mr. Pang Chung Khiang, Mr. Pore Ghee Lye, Dr. Pee Eng Yau and Mr. Wong Ka-Yoon, in the recent effort to update this framework.

References

1. Pang, C.K., Sim, K.W., Koh, H.S.: Evolutionary Development of System of Systems through Systems Architecting. In: Tan Y.H. (ed.) DSTA Horizons 2012, pp. 90–103. DSTA, Singapore (2012)
2. Kang, S.C., Pee, E.Y., Sim, K.W., Pang, C.K.: Framework for Managing System-of-Systems Ilities. In: Tan Y.H. (ed.) DSTA Horizons 2013/14, pp. 56–65. DSTA, Singapore (2013)
3. de Weck, O., Roos, D., Magee, C.: Engineering Systems: Meeting Human Needs in a Complex Technological World, pp. 65–96. MIT Press, Cambridge (2012)
4. Systems Engineering for Mission Assurance. In: MITRE Systems Engineering Guide, pp. 155–157. The MITRE Corporation (2014). http://www.mitre.org/publications/systems-engineering-guide/enterprise-engineering/systems-engineering-for-mission-assurance
5. U.S. Department of Defence, Defence Acquisition University (DAU): Robust Design. In: Glossary of Defence Acquisition Acronyms and Terms, 15th edn. DAU Press, Virginia (2012). https://dap.dau.mil/glossary/pages/2603.aspx
6. INCOSE Resilient Systems Working Group: Working Definition of Resilience. http://www.incose.org/docs/default-source/wgcharters/resilient-systems.pdf?sfvrsn=6 (2011)
7. de Weck, O.L., Ross, A.M., Rhodes, D.H.: Investigating relationships and semantic sets amongst system lifecycle properties (ilities). In: 3rd International Engineering Systems Symposium, CESUN, TU Delft, Netherlands (2012)
8. Richards M.G., Hastings D.E., Rhodes D.H., Ross A.M., Weigel A.L.: Design for survivability: concept generation and evaluation in dynamic tradespace exploration. In: 2nd International Symposium on Engineering Systems, CESUN. MIT, Cambridge (2009)
9. Mekdeci, B., Ross, A.M., Rhodes, D.H., Hastings, D.E.: Examining Survivability of Systems of Systems. In: 2011 INCOSE International Symposium, pp. 569–581. Wiley, Denver (2011)
10. Heinimann, H.R.: Future Resilient Systems. Presentation Slides, 3rd Cities Roundtable, Singapore. www.clc.gov.sg/documents/books/Future%20Resilient%20Systems.pdf (2014)
11. Hollnagel, E.: Resilience Engineering. http://erikhollnagel.com/ideas/resilience-engineering.html (2015)