Considering Alternative Strategies for Value Sustainment in Systems-of-Systems

Nicola Ricci, Adam M. Ross, Donna H. Rhodes, Matthew E. Fitzgerald

IEEE SysCon 2013
April 17, 2013
Orlando, FL
Outline

• Motivation
• Value Sustainment Strategies
  – Self-recovery
  – Change
  – Evolve
• Era Analysis
• Maritime Security SoS Case Study
• Discussion
MOTIVATION
Continually Changing World

The world in which Systems Engineers practice has undergone a significant metamorphosis over the past twenty years

- **Advent of the internet** → great increase in amount of resources available
- **Information travels at the speed of light** → instantaneous communication
- **High-speed computation** → Performance of very complex analyses

Systems are subject to highly dynamic operational environments

- A multitude of exogenous uncertainties can impact a system
  - **Geo-political shifts** (e.g., policy/regulation changes)
  - **Disruptive technologies** (e.g., advent of GPS)
  - **Market variations** (e.g., price &demand variations)

- Unanticipated shifts in stakeholder needs
  - Change of preferences
  - Change of mission objectives

RISK v. OPPORTUNITY

If focused solely on the present state of the world, engineers may encounter the problem of designing systems that may not be well positioned in the future, forced to operate in contexts for which they were not conceived, and delivering capabilities no longer of interest to stakeholders.
The purpose of any system is to provide some level of value to its stakeholders. A fighter jet may provide value to its user by exerting force on enemies or serving as a deterrent. The interest of the stakeholders is that the system keep providing value throughout their expected lifecycle, despite the occurrence of perturbations.

- Context changes: perturbations like jamming or thunderstorm can affect the value delivery of the fighter jet.
- Needs changes: a fighter jet is suddenly expected to be undetectable by radar.

System designers attempt to make systems that can provide value in spite of these perturbations. They can design an overly stable aircraft for any thunderstorm or include the option of a thunderstorm-targeted control system the pilot can activate.
Systems of Systems

- **Five characteristics** that distinguish SoSs from traditional systems (Maier, 1998)
  - Operational Independence of Elements
  - Managerial Independence of Elements
  - Evolutionary Development
  - Emergent Behavior
  - Geographic Distribution

- **Traditional SE v. SoSE**
  - Activities performed vary – SoSE more dynamic

- **Four different types of SoSs** (Dahmann et al., 2008)
  - **Directed**: built and managed to fulfill specific purpose (*focus of this paper*)
  - **Acknowledged**: constituent systems retain independent ownership/objectives/funding
  - **Collaborative**: constituent system interact to fulfill agreed upon purpose
  - **Virtual**: lack of central authority and agreed upon purpose

Most Systems of Systems do undergo periodic re-architecting – to respond or anticipate changes in needs and/or context – but this does not necessarily occur at the “speed of need”.
The Need to Cope with Change

The question for architects in directed SoSs, who are making system-level decisions during SoS inception or during a periodic re-architecting activity, is:

What value sustainment strategies will be most appropriate?

- What changes will prove to be unnecessary?
- What necessary changes should occur with greater ease and at less cost?

This paper proposes a framework for evaluating and comparing different value sustainment strategies over the lifecycle of the SoS

- During (re)design, architects must think of ways to appropriately respond to perturbations
- Mitigate risk (downside uncertainty)
  - Ex.: Increase UAV altitude in face of enemy attack
- Exploit opportunities (upside uncertainty)
  - Ex.: Adding new UAV type to leverage emerging technologies

“In my opinion the two things most adverse to good counsel are haste and passion” (Thucydides, ~400BC)

SoS Engineering is a dynamic process, where the identification of salient strategies for dealing with external changes can significantly benefit engineers and architects
VALUE SUSTAINMENT STRATEGIES
Wave Model

• Time-sequenced implementation of the SoSE process
• Wave model* reflects SoS attributes
  • Multiple overlapping iterations of evolution (incremental development)
  • Ongoing analysis (unlike traditional systems)
  • Continuous input from external environment
  • Architecture evolution

Main elements
  • Initiate SoS
  • Conduct SoS analysis
  • Develop/evolve SoS architecture
  • Plan SoS update
  • Implement SoS update
  • Continue SoS analysis

The wave model incorporates the dynamic nature in which SoS Engineering is performed

*Dahmann et al., 2011
Architecture-Design-System* Construct

LEVEL OF ABSTRACTION

Architecture
- Defined by a set of characteristics that SoS architects deem fundamental to specifying allowed forms, functions and behaviors of the SoS

Design
- Specific instance of parent architecture
- Changing between design instances within an architecture is usually less challenging and time-consuming than changing from one architecture to another

Changes in design and changes in architecture correspond to different levels of effort and impact on value delivery

*Beesemyer, 2012
= Perturbation
= Utility-enhancing opportunity
$A_n$ = n\textsuperscript{th} Architecture

Change Type:
\(\delta\) = Change in design (within pliable set)
\(\Delta\) = Change in architecture

Initiate SoS

Monitoring and Analysis

Operations

SELF-RECOVER

Time

Generation Length

$A_1$

$A_2$

$A_n$
Value Sustainment Strategy #1: Change

- Related ility: CHANGEABILITY
  - ability of a system to alter its form—and consequently possibly its function—at an acceptable level of resource expenditure
  - Different types: flexibility, adaptability, scalability, modifiability
  - Agent-Mechanism-Effect framework [Ross and Rhodes, 2008]
    - Agent → instigator
    - Mechanism → enabler
    - Effect → change of state
  - Enabled by inclusion of change options (path enabler + mechanism)

- Examples of changeability
  - Having a modular payload bay enables swapping the payload, and thus be changeable when facing different needs
  - High-altitude UAVs enables the possibility of changing to avoiding physical attacks

“CHANGE” strategy → changes in design
Value Sustainment Strategy #2
Evolve

- Relatedility: **Evolvability**
  - *The ability of an architecture to be inherited and changed across generations [over time]*
  - Design principles for evolvability

### Design Principle | Implications for Evolvability
---|---
Targeted Modularity | Limits change propagation (Hansen 2003) (Holts-Otto 2005)
Integrability | Compatibility and common interfaces (Fricke and Schulz 2005)
Scalability | Of a parameter or entire system to meet new needs (Fricke and Schulz 2005)
Decentralization | Distributed resources to limit effect of changes (Fricke and Schulz 2005)
Redundancy | Gives flexibility to designer to eliminate components (Fricke and Schulz 2005)
Reconfigurability | Self similar parts and maximizing information reconfiguration (Siddiqi and de Weck 2008)
Leverage Ancestry | Successful design choices from all prior generations (Kelly 2010)
Mimicry | Successful design choices from other systems/domains (Kelly 2010; Henderson and Clark 1990)
Disruptive Architectural Overhaul | Upgrading large aspects of architecture at a time (Kelly 2010; Henderson and Clark 1990)
Resourceful Adaptation | Repurposing successful design choices from other systems (Kelly 2010)

*(Beesemyer, 2012)*

Evolution of SoSs over time

“EVOLVE” strategy ➔ changes in architecture
Value Sustainment Strategy #3
Self-Recover

- Related ilities: **SURVIVABILITY (1)** & **ROBUSTNESS (2)**
  - (1) ability of a system to minimize the impact of a finite duration **disturbance** on value delivery
  - (2) ability of a system to maintain value delivery in spite of long-term changes [**shifts**] in needs or context
- Enabled by inclusion of **resistance options** (path inhibitor + resistance mechanism) and **change options** (path enabler+ change mechanism)

An **epoch** is a discrete time period where **expectations** and **context** are fixed.
ERA ANALYSIS
Using Epochs for Proposing Time-Based Strategies across Eras

Discretization of change timeline into short run and long run enables analysis

**Epoch:**
Time period with a fixed context; characterized by static constraints, design concepts, available technologies, and articulated needs

**Era:**
Time-ordered sequence of epochs

In order to pursue dynamic strategies, a system must have temporal properties, e.g., “-ilities” such as flexibility, evolvability, or survivability

(Ross and Rhodes 2008)
Era Simulation and Analysis

Era Analysis uncovers additional information, emergent only when considering time-ordered effects of uncertainty across the system’s lifetime

- Simulation of sample eras (constructed by sequencing epochs according to some model) allows collection of more data
  - Change mechanism usage
  - Cost/benefit “going rates” for adding/removing changeability
  - Lifetime cost/utility/revenue/efficiency statistics

Conceptually: “how does this system perform, using various value sustainment strategies, when uncertainty evolves over time?”
CASE STUDY
High-level Operational Needs Statement:

*Provide maritime security for a particular littoral Area of Interest (AOI)*

Stakeholders want a system (SoS) that:

- Detects, identifies and boards boats entering AOI
- Is capable of carrying out search and rescue missions upon request

**Architecture & Design Variables**

**Form**
- Hermes
- Shadow
- Prop Plane
- Helicopter
- Manned Patrol Boat
- Satellite Relay
- Land Sensors

**ConOps**
- Tech Level Upgrade
- Info Sharing Use
- Task Assignment
- Geographic Segmenting
- Operators Per UAV
- Workforce Buffer
- Authority

**Perturbations**

**Disturbances**
- Serious Attack Occurrence
- Asset Unavailable
- Information Attack
- Storm
- Tsunami

**Shifts**
- Technology Level
- Workforce Availability
- Info Sharing Availability
- Boat Arrival Rate
- Pirate Percentage
- Smuggler Percentage
- Search and Rescue
- Jamming (Bad Com)
Eight designs considered for evaluation in Era Analysis

- Each design is associated with a specific *initial architecture*
- Previously selected among the entire design space using robustness and survivability *screening metrics* (Fitzgerald, 2012)

<table>
<thead>
<tr>
<th>Design ID</th>
<th>Rationale for selection</th>
<th>Design Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Robust under preference set 1</td>
<td>Hermes: 2, Shadow: 2, Prop: 0, Helo: 0, Boats: 0, Task Assign.: 4, Zones: 2, Operators: 2:1, Authority: Central, Workforce Buffer: 0%</td>
</tr>
<tr>
<td>B</td>
<td>Robust under preference set 1</td>
<td>Hermes: 2, Shadow: 2, Prop: 0, Helo: 0, Boats: 0, Task Assign.: 4, Zones: 2, Operators: 2:1, Authority: Central, Workforce Buffer: 0%</td>
</tr>
<tr>
<td>C</td>
<td>Robust under preference set 2</td>
<td>Hermes: 2, Shadow: 2, Prop: 0, Helo: 0, Boats: 0, Task Assign.: 4, Zones: 2, Operators: 2:1, Authority: Central, Workforce Buffer: 0%</td>
</tr>
<tr>
<td>D</td>
<td>Robust under preference set 2</td>
<td>Hermes: 2, Shadow: 4, Prop: 0, Helo: 0, Boats: 0, Task Assign.: 4, Zones: 2, Operators: 2:1, Authority: Central, Workforce Buffer: 0%</td>
</tr>
<tr>
<td>E</td>
<td>Survivable to some of the perturbations considered</td>
<td>Hermes: 2, Shadow: 4, Prop: 0, Helo: 0, Boats: 0, Task Assign.: 4, Zones: 2, Operators: 2:1, Authority: Distr., Workforce Buffer: 0%</td>
</tr>
<tr>
<td>F</td>
<td>Very robust within 1% of the Pareto frontier</td>
<td>Hermes: 2, Shadow: 6, Prop: 0, Helo: 0, Boats: 0, Task Assign.: 4, Zones: 2, Operators: 2:1, Authority: Central, Workforce Buffer: 0%</td>
</tr>
<tr>
<td>G</td>
<td>Robust across all preference sets and cost types, in specific context of interest</td>
<td>Hermes: 6, Shadow: 6, Prop: 0, Helo: 0, Boats: 0, Task Assign.: 12, Zones: 2, Operators: 2:1, Authority: Central, Workforce Buffer: 0%</td>
</tr>
<tr>
<td>H</td>
<td>Expert opinion (exploration into large designs with workforce buffers)</td>
<td>Hermes: 6, Shadow: 6, Prop: 2, Helo: 2, Boats: 12, Task Assign.: 1, Zones: 2, Operators: 2:1, Authority: Central, Workforce Buffer: 33%</td>
</tr>
</tbody>
</table>

Initial designs are then allowed to change to any other accessible design point in the design space throughout the era *(depending on strategy adopted)*
DISCRETE-EVENT SIMULATION (Mekdeci, 2013)

- Used to evaluate the performance of different MarSec SoS design instances in given epoch
- Allowed inclusion of various stochastic processes (e.g., boat arrival rate)
- Allowed analysis of SoS emergent behaviors arising from the interaction of constituent systems and operational choices (e.g., how the presence or number of particular UAVs interact with the task management strategy adopted)
- Evaluated 10,368 alternative designs and 128 alternative contexts
- Was verified and validated
- Evaluated performance was used to generate SoSs’ utility scores

MarSoS DES is used to evaluate SoS performance in given epoch
ILLUSTRATIVE ERA (8 years)

- For the purpose of performing Era Analysis over the long run for the SoS
- Favorable and unfavorable *epoch shifts* alternate as the SoS operates through time

<table>
<thead>
<tr>
<th>Epoch #</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epoch ID</td>
<td>11</td>
<td>27</td>
<td>19</td>
<td>20</td>
<td>55</td>
<td>111</td>
<td>110</td>
</tr>
<tr>
<td>Epoch Duration (months)</td>
<td>12</td>
<td>6</td>
<td>24</td>
<td>6</td>
<td>24</td>
<td>18</td>
<td>6</td>
</tr>
</tbody>
</table>

| Tech Level | Low | Low | Low | Low | High | High | High |
| Workforce  | 100%| 100%| 100%| 100%| 100% | 67%  | 67%  |
| Info Sharing | Off | On  | On  | On  | On   | Off  | Off  |
| Boat Arrival | 1/640sec | 1/640sec | 1/320sec | 1/320sec | 1/320sec | 1/640sec | 1/640sec |
| Smuggler Percentage | 5% | 5% | 5% | 5% | 5% | 5% | 1% |
| S&R       | No  | No  | No  | No  | Yes | Yes | Yes |
| Jamming  | No  | No  | No  | Yes | No  | No  | Yes |

Era Analysis allows for the evaluation of different strategies over the course of the given era.
**Transition rules**

1. Reduce to pre-validated vehicle set (design-level change), i.e. expert-picked stable designs, which can be implemented with current operating SoS constituent systems.

2. Changing short-term (design-level change) ConOps (i.e., Task Assignment, Operators per UAV).

1. Changing long-term (architecture-level change) ConOps (i.e., number of Zones, Authority).

2. Adding Constituent Systems (architecture-level change). The cost of adding is the cost of the new vehicles, and the delay of adding is vehicle-dependent.

Path enablers for changeability can affect the cost and schedule associated with transitions:

1. Stockpiles of UAVs for quicker transitions
2. Up front training to lower cost and delay of changing CONOPS
No changes allowed

No re-architecting

1 re-architecting
(4 years in)

3 re-architecting
(2, 4 & 6 years in)
# Lifecycle Results for “Change” Strategy

**“Change”**

**No re-architecting**

<table>
<thead>
<tr>
<th>Design</th>
<th>Total Utility - Security (utile-months)</th>
<th>Total Discounted Cost ($100M)</th>
<th>Total Down-Time (months)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>44.8</td>
<td>2.33</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>44.7</td>
<td>2.33</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>44.7</td>
<td>2.35</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>44.9</td>
<td>2.33</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>27.2</td>
<td>2.51</td>
<td>3</td>
</tr>
<tr>
<td>F</td>
<td>44.8</td>
<td>2.26</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>46.4</td>
<td>2.82</td>
<td>0</td>
</tr>
<tr>
<td>H</td>
<td>58.0</td>
<td>3.05</td>
<td>0</td>
</tr>
</tbody>
</table>

Similar utility performance amongst designs of interest, except for E (poor) and H (excellent, but at ~25% cost increase)

Maximize efficiency approach is used for design changes between epochs
Lifecycle Results for “Self-Recovery” Strategy

<table>
<thead>
<tr>
<th>Design</th>
<th>Total Utility – Security (utile-months)</th>
<th>Total Discounted Cost ($100M)</th>
<th>Total Down-Time (months)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>+0</td>
<td>+0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>+13.5</td>
<td>+0.22</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>+14.6</td>
<td>+0.33</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>+14.7</td>
<td>+0.38</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>-25.3</td>
<td>+1.74</td>
<td>+87</td>
</tr>
<tr>
<td>F</td>
<td>+17.1</td>
<td>+0.86</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>+7.5</td>
<td>+4.21</td>
<td>+24</td>
</tr>
<tr>
<td>H</td>
<td>+10</td>
<td>+4.70</td>
<td>0</td>
</tr>
</tbody>
</table>

- General increase in utility/cost implies that max efficiency strategy is reducing the size of the designs
- Tradeoff appears likely to be “worth it” for already-small designs (+25% utility, +10% cost) but less so for larger designs (+15% utility, +100% cost)
- Not changing puts you at risk of more downtime
- Possibly an alternative strategy would be best for the large designs
# Lifecycle Results for “Evolve” (Once) Strategy

**“Self-recover”**
- No changes allowed

**“Change”**
- No re-architecting

**“Evolve”**
- 1 re-architecting

<table>
<thead>
<tr>
<th>Design</th>
<th>Total Utility – Security (utile-months)</th>
<th>Total Discounted Cost ($100M)</th>
<th>Total Down-Time (months)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>+0 44.8  +0 2.33  +0 0 0 +0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>+13.5 44.7  +0 2.33  +0 0 0 +0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>+14.6 44.7  +0 2.35  +0 0 0 +0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>+14.7 44.9  +0 2.33  +0 0 0 +0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>-25.3 27.2  +5.4 2.51  +0.17 87 3 +0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>+17.1 44.8  +0 2.26  +0 0 0 +0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>+7.5 46.4  +0 2.82  +0 24 0 +0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>+10 58.0  +0 3.05  +0 0 0 +0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Only E benefits from single rearchitecting over none
- Decisions about rearchitecting plans to be made weighing the pros and cons of additional/fewer architecting efforts
## Lifecycle Results for “Evolve” (3 Times) Strategy

### Summary

- **“Self-recover”**
  - No changes allowed

- **“Change”**
  - No re-architecting

- **“Evolve”**
  - 1 re-architecting

- **“Evolve”**
  - 3 re-architecting

### Design Results

<table>
<thead>
<tr>
<th>Design</th>
<th>Total Utility – Security (utile-months)</th>
<th>Total Discounted Cost ($100M)</th>
<th>Total Down-Time (months)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>+0 44.8 +0 +0 +0 2.33 +0 +0 0 0 +0 +0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>+13.5 44.7 +0 +2.1 +0.22 2.33 +0 -0.18 0 0 +0 +0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>+14.6 44.7 +0 +15.9 +0.33 2.35 +0 -0.09 0 0 +0 +0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>+14.7 44.9 +0 +14.8 +0.38 2.33 +0 -0.07 0 0 +0 +0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>-25.3 27.2 +5.4 +11.9 +1.74 2.51 +0.17 -0.39 +87 3 +0 +0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>+17.1 44.8 +0 +15.9 +0.86 2.26 +0 -0.08 0 0 +0 +0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>+7.5 46.4 +0 +10 +4.21 2.82 +0 +0.09 +24 0 +0 +0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>+10 58.0 +0 +1.6 +4.70 3.05 +0 +0.18 0 0 +0 +0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- All except Design A benefit from the triple re-architecting, even to the point of gaining utility AND decreasing cost.
DISCUSSION
• Analysis **demonstrative** in nature
• General understanding of the *contextual and needs-related uncertainties* affecting the SoS, as well as the *possible strategies* one can employ to respond to how uncertainty unfolds, are **prerequisites** for the application of the analysis presented
• Addresses some of the **important issues** that characterize modern-day SoSE
  • Considering (and possibly highlighting options to facilitate the implementation of) possible timed and/or contingent SoS changes **early in the design phase**
    • Ex.: Include multi-role asset to respond to loss of UAV via re-tasking
    • Design E performs badly when it can’t rapidly and effectively change
  • The value of a given strategy (self-recover, change or re-architect) is dependent on the **initial design selected**
    • Ex.: “change” strategy “addition of back-up assets” to the operating SoS is more effective for smaller (in number of assets) initial SoS designs
• Consideration of the **timing of execution of architectural changes**
  • re-architecting once was hardly beneficial for any design; re-architecting three times nearly benefitted all designs (improved performance)
Wrapping up

• **Different strategies** can be considered, as well as **alternative eras**
• **Validation** of the approach
• Strategy selection for a particular design is dependent on the era considered
• A “**multi-era**” **analysis** could be the basis for future research in terms of helping architects to select design-strategy pairs that are most robust across different likely era progressions

Overall contributions of the analysis:

• Era Analysis enables the **exploration** and **comparison** of results associated with the adoption of **alternative “change strategies”**
• Helps systems architects **choose** an appropriate value sustainment strategy for a given envisioned era, as well as **highlight salient time-scales** for SoS intervention
• Helps identify changes that will turn out to be **unnecessary**, or others that will prove to be very important and need to occur with **greater ease**
• Invites systems architects to **think about the adoption of different strategies before the inception of the operational SoS**
QUESTIONS?

SELF-RECOVER

CHANGE

Plan δ

Implement δ

EVOLVE

Re-architect

Plan Δ

Implement Δ