Interoperable Acquisition Overview

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA  15213

Jim Smith

September 11, 2008
Purpose

1. Introduce key principles of **interoperable acquisition**
2. Explore their application in a systems of systems context
3. Examine potential system of systems impacts to a single program
Overview

System-of-System Fundamentals

Interoperable Acquisition

Impacts in Practice

Summary
Overview

System-of-System Fundamentals

Interoperable Acquisition

Impacts in Practice

Summary
Essential Characteristics of Systems of Systems

Autonomous constituents with independent operations and management

- Includes people, organizations, software agents, etc.
- Source of independent actions and decisions

Independent evolution of each constituent

- Can respond to new technology and mission needs at its own pace and direction

Emergent behavior

- “Whole is different than the sum of the parts”
- Indirect and cumulative effects of influences, actions, interactions

Must recognize, manage, and exploit the inherent nature of these characteristics
Systems of Systems Involve Multiple Perspectives

Integration and interoperability requires orchestration of a diverse set of stakeholders and multiple organizations.

Management Perspective
- Time-phasing of deliverables
- Effects of delays
- Funding and budget
- Risk management
- Multi-supplier monitoring
- Multi-supplier coordination
- etc.

Development/Assembly Perspective
- Capabilities available (from other systems)
- Architecture
- Development-based AND assembly-based construction
- Testing
- etc.

Operational Perspective
- Operational stakeholder needs
- Concept of operations
- Deployment and support
- etc.

Integration and interoperability requires orchestration of a diverse set of stakeholders and multiple organizations.
## Why Isn’t This “Just” a Scaling Issue?

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Traditional Systems</th>
<th>Systems of Systems</th>
<th>Essential Characteristic</th>
</tr>
</thead>
</table>
| Constituents | All constituents are known and visible | Changing and potentially unknown constituents  
  - Entity assembling system of systems may not know constituents until run-time  
  - Constituent may not know it is part of a system of systems | Autonomous constituents                  |
| Purpose  | Predetermined by system owner and conveyed to constituents | Continuously evolving  
  - Cooperatively determined  
  - May or may not be known by systems participating in system of systems |  
  - Autonomous constituents  
  - Independent evolution  
  - Emergent behavior |

*From a presentation by Brownsword, L. and others "Overview of System of Systems Navigator"*
Why Isn’t This “Just” a Scaling Issue?

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Traditional Systems</th>
<th>Systems of Systems</th>
<th>Essential Characteristic</th>
</tr>
</thead>
</table>
| Control      | Hierarchically-structured with central control by system owner                       | • Constituents do not control how & when their systems are used in the system of systems (but may have control over their systems)  
• Entity assembling system of systems has control over assembly but not over the participating systems | Autonomous constituents           |
| Requirements | Defined and managed by system owner                                                  | Systems participating the system of systems often have to anticipate how their system will be used | Emergent behavior                |
Why Isn’t This “Just” a Scaling Issue?

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Traditional Systems</th>
<th>Systems of Systems</th>
<th>Essential Characteristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boundaries</td>
<td>Clearly defined boundaries</td>
<td>Unclear and evolving boundaries</td>
<td>Emergent behavior</td>
</tr>
<tr>
<td>Visibility</td>
<td>All aspects can be seen, understood, and controlled</td>
<td>No one stakeholder or group can have complete, global insight across the entire system of systems</td>
<td>Autonomous constituents</td>
</tr>
</tbody>
</table>
Influence Relationships

System of systems constituents influence—and are influenced by—each other

The *indirect* and *hidden* influences of systems of systems are not present in single systems, and are the primary mechanism behind emergent behavior

Influence relationships arise from many sources:

- Contract language
- Technical requirements
- Reporting requirements
- “Giver/receiver”
- Funding
- Individual trust

Influence is like gravity: it holds the system of systems together

Representative System of Systems Context Diagram
Indirect and Cascade Influence Relationships

Relationships exist where constituents influence one another

Sequences of direct neighbor interactions often generate indirect effects between distant constituents

Indirect effects often form cascades

• Detailed steps often unpredictable and difficult to envision
• Cumulative effects can be predictable
• Emergent effects define character and utility of resulting system of systems

System of systems risks may not be apparent for individual constituents, or by analyzing only “near neighbor” interactions

• $S_2$ has a backwards compatibility relationship with $S_1$
• $S_3$ has a schedule dependency on $S_2$
• Therefore, $S_1$ and $S_3$ are indirectly related through $S_2$
Overview

System-of-System Fundamentals
Interoperable Acquisition
Impacts in Practice
Summary
Interoperability

Traditionally, interoperability has been viewed as the ability of operational systems to exchange data or communicate in some defined way.

This definition focuses entirely on the “tail end” of the process, ignoring all that must first happen to make operational interoperability possible.

A better definition of interoperability is:

*The ability of a set of communicating entities to:

1. exchange specified information and
2. operate on that information according to a specified, agreed-up, operational semantics.*

This permits the discussion of different aspects of interoperability, such as the management, construction, and operation of a system.
Conventional notions of “point-to-point” interoperability is insufficient in a system of systems context: interoperability requirements cannot be completely specified during construction, only becoming apparent at run time … and they may change each and every time!

Individual systems ("A," "B," and "C")

System “D” connected to “A,” “B,” and “C”

System-of-systems “X” emerges from the addition of “D”
Current Approaches to Interoperability

Current approaches to providing capability based on integration of and interoperation amongst existing systems

- Constituent systems not necessarily developed with “broad” interoperability in mind
- Acquisition (development, sustainment, etc.) focused on their individual systems/programs, with their individual requirements, funding, users, etc.
- Integration “after the fact”
Limitations With Current Approaches to Interoperability

Disconnect between the goals and reality

Goal:
- Seamless interoperability
- Dynamic, flexible, responsive to changing environment and threat, survivable
- Programs will “cooperate and graduate”

Reality:
- Interoperability either insufficiently defined (e.g., “interoperable with XYZ system”) or too narrowly constrained (e.g., “interoperable with XYZ system, using TDMA over 5 KHz UHF DAMA SATCOM …” etc.
- Integration results in brittle systems that are fragile and difficult to sustain
- Stovepiped programs/systems not incentivized to engage in altruistic behavior
Acquisition

For our purposes, *Acquisition* (with a “Big ‘A’”) encompasses all the processes used to develop, procure, deploy, employ, and sustain systems throughout their complete lifecycle.
To achieve operational interoperability, you must also consider the programmatic and constructive aspects of interoperability.
Interoperable Acquisition

Given the all-encompassing nature of acquisition, and what it means to be interoperable, leads to a definition for interoperable acquisition:

- The set of practices that enable acquisition, development, and operational organizations to more effectively collaborate to field interoperable systems.
- This is achieved through sharing relevant information and performing necessary activities that enable the collective behavior of these organizations to successfully deliver system-of-systems capabilities.
Interoperable Acquisition Context
Aspects of Interoperable Acquisition

Programmatic, constructive, and operational interoperability are aspects of interoperable acquisition.

Most real problems must address the interaction among the various aspects. For example:

- **Cost** (programmatic) may affect the **architecture** (constructive) which affects the capabilities of an **operational system** (operational).
- **Technology maturity** (constructive) may affect the **risk** of holding **schedule** (programmatic) to deliver an **operational system** (operational).
- The ability to reflect evolving **doctrinal concepts** (operational) into a **system of systems acquisition strategy** (programmatic) and **system of systems architectural concepts** (constructive)
Interrelationships Between Interoperability Aspects
Overview

System-of-System Fundamentals
Interoperable Acquisition
Impacts in Practice
Summary
Impacts in Practice

How can this understanding of the differences between systems of systems and system-centric thinking be put to use?

• Avoidance of common mistakes
• Attributes of “good” practices in a system of systems context
Common Pitfalls

What are the dangers in applying systems- and program-centric thinking in a systems of systems context? Some examples include:

• Managing risk at the individual program level
• Optimizing system engineering at the individual system level
• Assuming that how a system will be employed is fixed and unchanging
• Top-down, hierarchical management
System-Centric Risk Management

Failure to apply risk management at the system of systems level

- Individual program/system risk management efforts “stepping on each others toes”
- Locally-optimized solutions can lead to serious degradation—or even total failure—at the system of systems level
- Emergent risks arise from the interaction of systems/organizations
System-Centric Systems Engineering

Failure to perform systems engineering activities at the system of systems level results in:

- Ineffective requirements, resource, schedule, etc. trades across multiple systems
- Sub-optimization at the system of systems level

Failure to recognize the programs/systems often exist in multiple sequential or simultaneous systems of systems contexts results in this effect extending across systems of systems

- Sub-optimization at the systems of systems level

Emergence gives rise to new behaviors
Disconnect Between Acquisition and Operations

Failure to understand the programmatics of the linkage between acquisition and operations results in significant disconnects between how systems are procured, fielded, and used

- Can’t be defined once and forgotten: doctrine, operational concepts, etc. constantly being revised
- Leads to loss of mission capability
- Significant cost—in time and effort—to effect “band aid” solutions

New operational concepts and doctrine can exhibit emergent behavior
System-Centric Programmatics

Rigid, hierarchical management and organizational patterns are inappropriate in systems of systems

The attempt to “bring order and discipline,” and to “eliminate emergent behavior” does neither

- Introduces significant—potentially fatal—time delays into the decision-making process
- Not all necessary information is globally available—nor should it be
- Lack of understanding about what “necessary information” really is means that decisions are still made on the basis of inaccurate and incomplete data … but it now takes longer!
Systems of Systems Practices and Processes

Program- and Systems-Centric Versus Systems of Systems

Practices and processes are determined within a single program

- Known causes and effects
- Dependencies are mainly in the context of the particular program or system
- Changing dependencies can be controlled
- Negotiation of information and decisions resides within the program

Practices and processes cannot be determined solely by a single program; influenced by everything else in the system of systems

- Causes and effects a combination of known, unknown, and unknowable
- Dependencies in context of the particular program associated with other programs. Changing dependencies may be managed
- Influences with other organizations and effects they could have on a particular acquisition program
- Control of information and decisions require negotiation to develop a shared understanding
Influence Management

Capture existing influence relationships

- Explicit understanding—and agreement—on the semantics of the relationship
  - Examples: What has to take place when? With what level of assurance? Degree of trust? For how much money? With what quality attributes?

- Define how things *actually happen* rather than how they are supposed to happen
  - Accuracy is important, but precision is not
  - Comprehensiveness is important, but completeness is impossible

Maintain current state of influence relationships

- The nature of—and the extent, parties, etc.—interdependency relationships will change

- Be aware of emergent, cascading, and multi-dimensional influence relationships
Schedule Management

Schedules should take into account external influences, constraints, etc.

- Make interrelationships explicit (tacit knowledge or assumed agreements may have drastic implications)
- Identify relationship chains that “cross over” from one aspect to another or have “downstream” effects

Control of schedule information and decisions require negotiation to obtain a consensus

- Global optimization may require sub-optimization at the individual project level

Schedules will continually change and evolve

Be pragmatic: There is no “perfect” schedule
Risk Management

Risk management goes beyond a common “green-yellow-red” format for reporting risks

- What is the impact on your project’s risk posture of the corresponding risks in other projects that may be part of a given SoS?
  - Factor in mitigation strategies/off-ramps, cost and schedule implications, etc.

Risk mitigation needs to factor in the broader SoS context

- Some mitigations may result in suboptimal results for individual systems
Rewards and Incentives

You get the behavior that you reward: organizational policies reflect the incentives, awards, and cultural norms of the “society” of practice

- Traditionally, rewards systems disincentivize altruistic SoS behavior

Align local reward systems with global operational mission or business objectives

Encourage—or require—innovation, altruism, etc.

- Seek out and reward innovative “out of the box” thinking
- Reward projects for making other projects’ jobs—and the broader SoS goals—easier
- Penalize projects (and contractors) for failing to disclose information that adversely impacts other projects and the SoS

Use emergence to good effect: small changes in rewards and incentives can have major payoffs!
Systems/Systems of Systems Engineering

Systems of systems engineering is about continually understanding and negotiating what capability can be provided that is sufficient to meet current and evolving operational or business objectives

- Continuous characterization of needs, of drivers of change, of evolving systems
- Control is no longer the purview of each individual project, but must be shared

Solutions are formed that encompass the full operational life

- Implications of funding, schedule, capability on transition to operations, support, repairs (and evolution)

Solutions are designed with uncertainty and change in mind

- Assume that the operational context for your system will be different than originally specified
- Assume that there may be unreliable communications (or unlimited bandwidth, zero latency, etc.)
- Assume that your system will be used with other systems in unplanned ways
Requirements Engineering

In a SoS context, an iterative approach to requirements is mandatory

• Impossible to “perfectly” specify requirements for complex systems
  – The user communities don’t know in detail what is really needed at the outset and their needs will change
  – The user communities and developer communities will discover problems interpretation of the requirements as you proceed
  – Solutions may be impractical and require changes to “requirements”

Prioritization of requirements in light of desired operational or business effects is crucial

• Pragmatically not all requirements can be met (and the list will continually change)
• Prioritization requires active participation from all relevant constituents

Requirements trade-offs across systems in a SoS-wide can’t optimize any one part at the expense of the SoS
Overview

System-of-System Fundamentals

Interoperable Acquisition

Impacts in Practice

Summary
Summary

Systems of systems require widespread interoperability

- Multiple system-of-systems operational contexts
- Multiple programs, services, agencies, etc.

Achieving this requires an all-encompassing view of acquisition: “The ‘Big A’”

- Understand the relationships between life cycle phases and interoperability aspects
- Necessary to appreciate the importance of shared understanding of desired operational capability, systems of systems perspectives, etc.

Rewards/incentives drive behavior

- Organizational architecture influenced by this “informal structure”
- Existing framework encourages system-centric behavior
Key Things to Remember

Top three “takeaways”:

1. Systems of systems are sufficiently different from traditional approach that it requires a different approach to acquisition

2. Scale does not simply mean that you just have to do “more” conventional engineering, management, or acquisition: scale brings about additional complexities in systems of systems that are not present in a program- and systems-centric environment

3. Acquisition must orchestrate program management, construction, and operational aspects both within a program, as well as across all constituents in a system of systems
DOCTOR FUN

The daydreams of cat herders

I think I'd rather manage a large software development project.

Used by permission of the artist
Some Recent Technical Reports

Meyers, C.; & Smith, J. “Programmatic Interoperability” (CMU/SEI-2008-TN-012)


Contact Information

Jim Smith
(703) 908-8221
jds@sei.cmu.edu
http://www.sei.cmu.edu/staff/jds/

ISIS Initiative
http://www.sei.cmu.edu/isis/