

# Evolution of a Software Engineer in a SoS System Engineering World

Software Engineering Institute  
Carnegie Mellon University  
Pittsburgh, PA 15213

Tricia Oberndorf, Carol A. Sledge, PhD  
April 2010



## NO WARRANTY

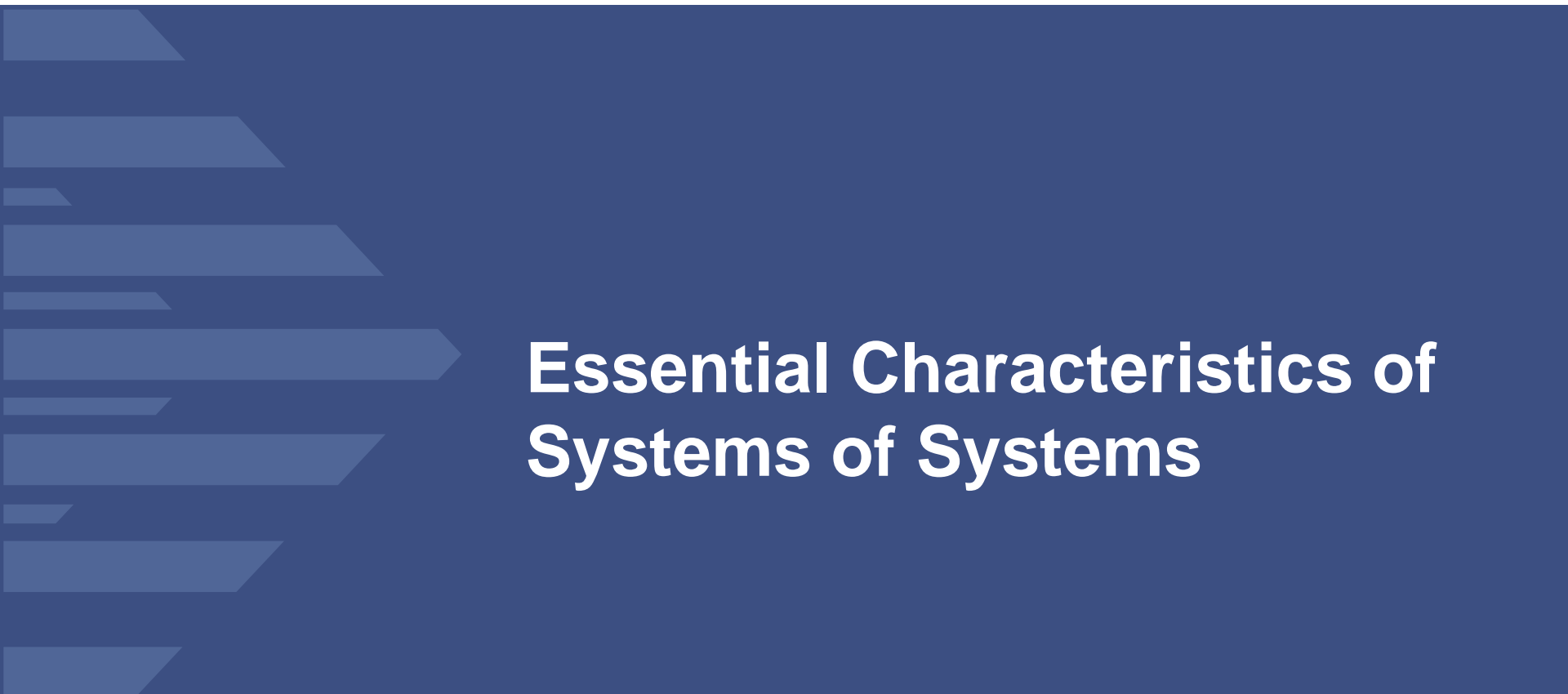
**THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.**

Use of any trademarks in this presentation is not intended in any way to infringe on the rights of the trademark holder.

This Presentation may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at [permission@sei.cmu.edu](mailto:permission@sei.cmu.edu).

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.





# Essential Characteristics of Systems of Systems



# Maier's Characterization of Systems of Systems

Autonomous constituents with independent operations and management

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

Evolution...

- Independent evolution of each constituent to respond to new technology and mission needs at its own pace and direction
- Evolution of the whole in response to changing demand

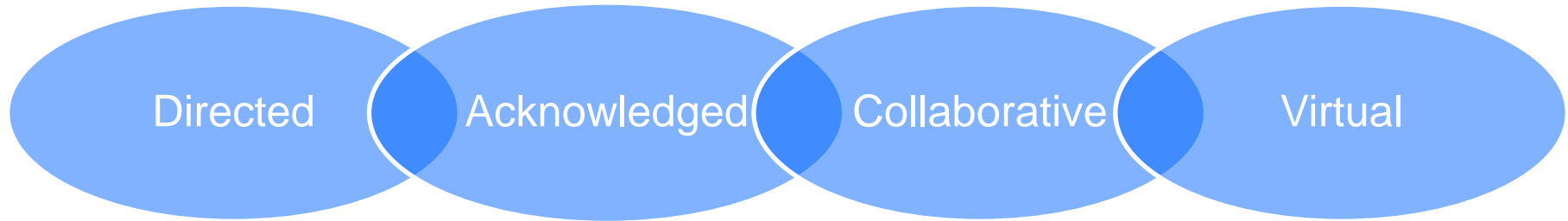
Emergent behavior

- “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***



# Types of SoS\*



- **Integrated SoS, built and managed to fulfill specific purposes**
- **Centrally managed to maintain and evolve**
- **Constituents independent but subordinated to centrally managed purpose**

- **Recognized objectives, designated manager and resources**
- **Constituents maintain independent ownership, objectives, funding, etc**
- **Changes based on collaboration between the SoS and the constituent**

- **Constituents interact more or less voluntarily to fulfill agreed central purposes**

- **Lack central management authority and centrally agreed purpose**
- **Rely on relatively invisible mechanisms to maintain it**

\* DoD System Engineering Guide for System of Systems Engineering (Version 1.0, August 2008) & Maier



# Development Challenges

**Increasingly complex net-centric systems-of-systems, with demands for responsiveness, fast turn-around for changes, and unprecedented flexibility.**

- Demands for increased integration, interoperability, flexibility, adaptability, more complex and dynamic capabilities
- Enterprise perspectives/requirements; sustainment concerns
- Software increasingly replacing hardware and connecting other systems (e.g., to reduce sensor to shooter loop)
- Software represents a major risk: Software is often the long pole in the tent.



# Similarities – Systems and SoS

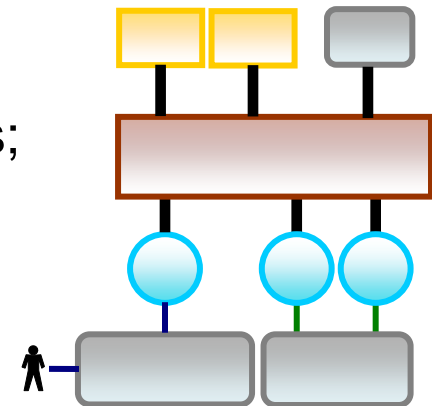
Multi-contractor teams using different processes; dispersed engineering, development & operational locations

New technologies create opportunities/challenges; products change/evolve, organizations mutate

Business/operational needs change - often faster than full system capability can be implemented

Skill set falls short of what is needed

Cost and schedule constraints limit options



***Many proven software development and management practices, methods, and tools exist for systems, but far too few projects take advantage of them.***



# Systems vs. Systems of Systems

Practices & processes are determined <i>within</i> a single program:	Practices & processes influenced by the whole SoS context:
Known causes and effects	Causes and effects a combination of known, unknown, and unknowable
Dependencies are mainly in the context of the particular program or system	Dependencies largely outside your span of control
Changing dependencies can be controlled	Changing dependencies <i>may</i> be managed; Influence is your only option
Negotiation of information and decisions reside within the program	Wider collaboration needed to develop shared understanding – or mitigation if negotiation doesn't work
Goal is system capabilities	Goal is SoS capabilities – constituent PLUS
Can focus on optimization of capabilities	Focus on <i>satisficing</i> for emergent capabilities
	New paradigms required due to: <ul style="list-style-type: none"> <li>• No central authority in general</li> <li>• Independent development and evolution of the constituents</li> <li>• Focus on needs at enterprise level</li> </ul>





# Implications for Engineers

We see a lot of these characteristics today, but in SoS they *dominate*.

The processes, artifacts, and collaborations in systems of systems are living and dynamic, not static.

- E.g., for an SoS, there is never a well-defined end-stage deployment — only constant evolution

Success with SoS is not simply a matter of doing more of the same on a larger scale

- Increased complexity, exponential growth of relationships, increase in cross-system and cross-organizational issues, etc.
- *Net-centricity and other paradigms for the future may be beyond the reach of today's software technologies and approaches*

***Rigor meets agility and innovation***



# For SoS, Some Principles Change



## Educate SoS staff on SoS software principles

- All software players must be cognizant of SoS software architecture.
- Other domain experts need a “reading level” understanding of SoS and software.

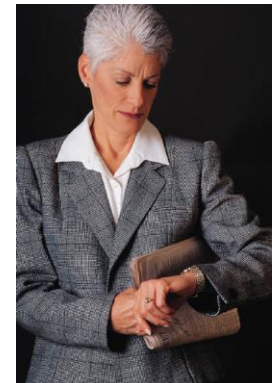
## SoS **mandates** use of good system & software engineering practices.

- Software (& system) engineering starts early in life-cycle
- Assurance work begins earlier in life-cycle AND continues throughout
- SoS engineers must know which “best practices” to apply



## SoS engineering demands

- Designated *skilled* leadership with responsibility AND authority
- Negotiated agreements with appropriate constituent management and system engineering counterparts

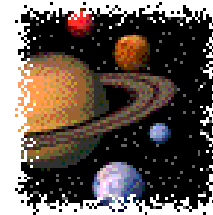


# Some Principles Are New

Constituents must be “self-contained” (“safe”) – no leaks, no undocumented behavior, all assumptions known (“formal methods lite”)

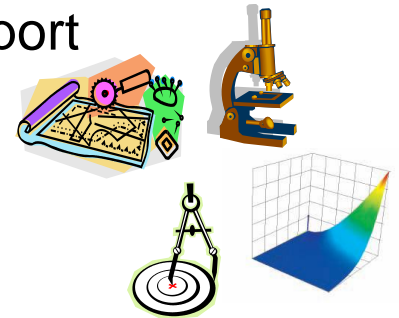
Intense involvement of true operational user

Constituents need to take responsibility for some run-time governance.



It is beyond the capacity of a human brain to grasp the whole SoS, so the kinds of analyses needed will not be possible without support

- Dictates use of a comprehensive set of systems and software tools for analysis, working together seamlessly



The SoS Chief Engineer needs to know when (not) to system engineer

- When to accommodate incompatibility, when to demand constituent changes?



# The SoS Engineer



# How Does This Change Life for the Engineer? -1

SoS engineers need:

- A new perspective
  - *Moving from the “mechanics of the parts” to the “dynamics of the whole”* (Sheard)
- To adjust their ideas of life-cycle
  - Design a little – program a little – integrate a little – test a little – *spiral*
  - Make extensive use of modeling and other non-testing analysis approaches
- To take risk management to a new level
  - Cross-cutting risks (across organizational boundaries and constituent systems): not just managed – also *balanced*
- A new requirements approach
  - Initial discovery of SoS requirements then continuous “rediscovery” as the SoS (constituents and whole) evolves – top-down AND bottom-up



# How Does This Change Life for the Engineer? -2

SoS engineers need:

- A new understanding of failure and recovery
  - What does it mean for a SoS to “fail”?
  - Can the SoS recover? Fast enough? Will roll back work?
- A new take on evidence and assurance methods
  - The processes, artifacts, and collaborations in SoS are dynamic and ongoing, not static: continual integration and test are necessary
  - New approaches to C&A are required
- To make deployment a continuous operation



# Prerequisites

Critical thinking skills (having and applying)

- Includes employment of appropriate tools: affinity diagrams, categorization, methods to evaluate alternatives, organizing tools, just to name a few

Broader vision (capacity for and willingness to apply)

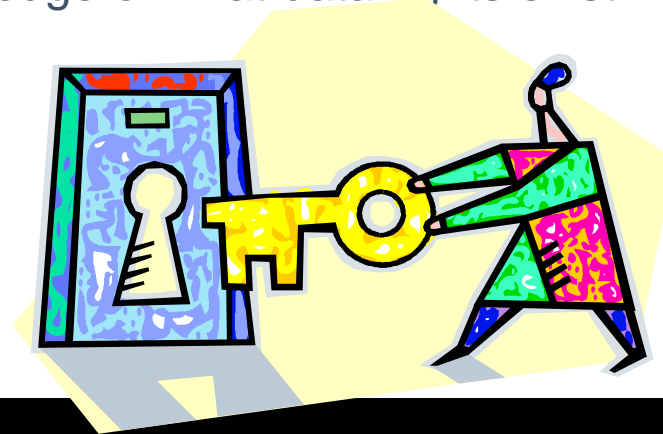
Broader educational background to support broader vision

People skills

- E.g., more emphasis on negotiation and coordination

Broader range of competencies (e.g., gets into legal and contractual)

- E.g., includes knowledge of what data rights exist



# New/Revised Roles -1

## Chief SoS Engineer

- Keeps overarching SoS perspective in mind, keeping right things connected
- Technical SoS risk management
- Coordinates with all constituent and all other Chief Engineers

## SoS Requirements Engineer

- Co-evolution of SoS CONOPS and SoS capabilities with constituent systems
- SoS requirements experimentation – design/program/integrate/test-a-little

## Chief SoS Software Architect

- Coordinate with constituent software architects
- Identifies SoS software quality attributes from critical mission threads

## Chief SoS Software Engineer

- Coordinate with constituent software engineers
- Participates in SoS-level trade-offs





# New/Revised Roles -2

## Chief SoS Analyst

- Employs modeling and simulation to examine SoS engineering trade-offs and alternatives and for gathering evidence for the assurance processes

## Chief SoS Assurance Engineer

- Develops evidence that SoS will behave as expected
- Oversees development and evolution of assurance cases
- Coordinates with constituent system engineers regarding impacts of changes on assurance arguments and evidence
- Oversees incremental demonstration of interoperability and SoS capability

## SoS Test Engineer

- Devises SoS tests, considering experimental test design and coordination with large-scale and joint exercises
- Advises Chief SoS Assurance Engineer when modeling will be more effective than test
- Engineers tests directly into software components for self-healthchecks



# New/Revised Roles -3

## Integrated SoS Engineering Environment Lead

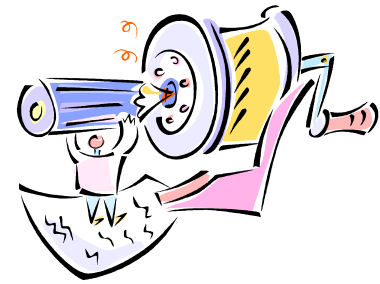
- Responsible for engineering environment for SoS development & evolution
  - Incorporates virtual, simulated, and actual SoS assets in support of life-cycle activities

## Senior SoS Technical Consultant/Mentor

- Senior organizational Fellow
- Consulting services & advice based on extensive personal SoS experience
- Mentors more junior architects and engineers regarding SoS knowledge and ability



# New Skills



- SoS knowledge codification & management
- Application of contract language for consistency and flexibility across the SoS
- Creation and management of relationships
- Trade-offs at unprecedented levels
- SoS strategic (holistic) thinking
- Negotiation, collaboration, listening, elicitation, facilitation, multi-disciplinary team focus
- SoS model-based engineering
- SoS analysis methods and tools
- Technology knowledge & assessment
- SoS methods for new risk management, requirements engineering, and test



# The Way Forward



Each skill we've listed/discussed is applicable to a range of engineers

- from junior ones (e.g., need to know what incentives are in a contract)
- to Chief SoS Engineers (e.g., need to frame and negotiate the contract language for incentives)

Some ideas:

- Lots of mentoring and on-the-job development\*, connected with the engineering through applied, hands-on learning
- Universities will need to strengthen the scientific and mathematical elements in their curricula
- More emphasis on SYSTEM engineering, including understanding how software fits into system and how to talk to system engineers
- Longer rotations\*
- Licensure could be considered in the future

\* Mennell, Ray. *Analysis of SEC's Continuing Professional Education*. Presentation to Army Strategic Software Improvement Program (ASSIP) Meeting, 18-19 February 2009.



# Potential Research Questions and Areas -1

Are our universities training/educating their graduates for this?

Are we offering today's professionals:

- Training in these new SoS (and other relevant) skills?
- Career paths that recognize the new skills for acquisition and development?

How do we make the required knowledge about paradigms and current SoS best practices available to the SoS architects, engineers, and acquirers?

What is an initial set of areas for which SoS best practices are needed?

- Such as SoS requirements engineering, SoS test engineering, SoS trade-off analyses, SoS modeling & analysis, SoS risk management

How do we codify requisite SoS knowledge and practices in the context of existing and emerging life-cycle processes?



# Potential Research Questions and Areas -2

For the SoS things we do NOT know how to do today:

- How do we collect the new best practices as they are discovered?
- How do we explore what we know we do not yet understand – the known unknowns?
- How do we protect against the unknown unknowns?



QUESTIONS?



# Contact Information

## Authors:

Patricia Oberndorf

Acquisition Support Program

Telephone: +1 412-268-6138

Email: [po@sei.cmu.edu](mailto:po@sei.cmu.edu)

Carol A. Sledge, Ph.D.

Research, Technology, and System  
Solutions

Telephone: +1 412-268-7708

Email: [cas@sei.cmu.edu](mailto:cas@sei.cmu.edu)

## World Wide Web:

[www.sei.cmu.edu](http://www.sei.cmu.edu)

## U.S. mail:

Software Engineering Institute

Carnegie Mellon University

4500 Fifth Avenue

Pittsburgh, PA 15213-2612

USA





# References

Lane, Jo Anne, Boehm, Barry. "System of Systems Lead System Integrators: Where Do They Spend Their Time and What Makes Them More or Less Efficient?", *Systems Engineering*, Vol. 11, No. 1, Feb. 2008, pp. 81-91.

M. W. Maier, "Architecting Principles for Systems of Systems," *Systems Engineering*, vol. 1, no. 4, pp. 267-284, 1998.

Office of the Secretary of Defense (OSD), DoD System Engineering Guide for System of Systems Engineering, vers. 1.0. Washington, DC: DoD, August 2008.

Sheard, S. Practical Applications of Complexity Theory for Systems Engineers. Systems and Software Consortium, Herndon, VA, 2005.

Weinstock, Charles B., Goodenough, John G., Hudak, John J. *Dependability Cases*. CMU/SEI-2004-TN-016, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA. May 2004.



# Who We Are



# Software Engineering Institute

Department of Defense R&D Laboratory FFRDC, created in 1984

Administered by Carnegie Mellon University

Headquartered in Pittsburgh, PA; offices and support worldwide

Mission - The SEI advances software engineering and related disciplines to ensure systems with predictable and improved quality, cost, and schedule.

## Stakeholders

- Government, commercial, and academic
- R&D and direct customer support

## Areas of Work

- Process
- Network security and survivability (CERT)
- Software/system architecture and SOA
- Systems of systems and net-centricity





**Software Engineering Institute**

**Carnegie Mellon**



**Software Engineering Institute**

**Carnegie Mellon**

Evolution of a Software Engineer  
Oberndorf and Sledge

© 2010 Carnegie Mellon University