

System of Systems V&V

John B. Goodenough

October 19, 2011



System of Systems

Constituents are operationally independent

- Provide functions that are useful in their own right

Constituents are managerially independent

- Separately acquired; have their own stakeholders

Constituent interactions create SoS value

- SoS performs functions, has properties, and carries out purposes not residing in any constituent itself (emergent behavior)

Continual evolution of constituents and SoS capabilities

Based on Maier, M.W., "Architecting Principles for System of Systems," Systems Engineering, Vol. 1, No. 4, 1998, pp. 267-284



V&V Implications

Constituents are operationally independent (**unexpected behavior**)

- Provide functions that are useful in their own right

Constituents are managerially independent

- Separately acquired; have their own stakeholders

Constituent interactions create SoS value

- SoS performs functions, has properties, and carries out purposes not residing in any constituent itself (emergent behavior)

SoS is continually evolving

Based on Maier, M.W., "Architecting Principles for System of Systems," Systems Engineering, Vol. 1, No. 4, 1998, pp. 267-284



V&V Implications

Constituents are operationally independent (**unexpected behavior**)

- Provide functions that are useful in their own right

Constituents are managerially independent (**change at own pace**)

- Separately acquired; have their own stakeholders

Constituent interactions create SoS value

- SoS performs functions, has properties, and carries out purposes not residing in any constituent itself (emergent behavior)

SoS is continually evolving

Based on Maier, M.W., "Architecting Principles for System of Systems," Systems Engineering, Vol. 1, No. 4, 1998, pp. 267-284



V&V Implications

Constituents are operationally independent (**unexpected behavior**)

- Provide functions that are useful in their own right

Constituents are managerially independent (**change at own pace**)

- Separately acquired; have their own stakeholders

Constituent interactions create SoS value (**behavioral interface**)

- SoS performs functions, has properties, and carries out purposes not residing in any constituent itself (emergent behavior)

SoS is continually evolving

Based on Maier, M.W., "Architecting Principles for System of Systems," Systems Engineering, Vol. 1, No. 4, 1998, pp. 267-284



V&V Implications

Constituents are operationally independent (**unexpected behavior**)

- Provide functions that are useful in their own right

Constituents are managerially independent (**change at own pace**)

- Separately acquired; have their own stakeholders

Constituent interactions create SoS value (**behavioral interface**)

- SoS performs functions, has properties, and carries out purposes not residing in any constituent itself (emergent behavior)

SoS is continually evolving (**constituent and SoS evolution**)

Based on Maier, M.W., "Architecting Principles for System of Systems," Systems Engineering, Vol. 1, No. 4, 1998, pp. 267-284



The SoS Perspective: Not the Usual Rules

Differences are unavoidable

- Requirements are only temporarily stable
 - Lack of agreement about the problem being solved
 - Continual change and renegotiation of needs
- Different groups want different configurations/services/etc.
 - And they change their minds!
- Tradeoff decisions are not stable
- Configuration information will not be accurate and cannot be tightly controlled
- Usage will drift from what was expected
- **Need to manage differences rather than attempt to eliminate them**

SoSs evolve continuously rather than discretely

- SoSs must be designed to tolerate the introduction of concurrent changes
- Which changes absolutely must be coordinated? What is the impact if miscoordination occurs? How is the impact detected, and mitigated?



Other Perspectives

Systems of systems (SoSs)

- Constituents are decisionally autonomous, independently interacting entities
 - Socio-technical ecosystems
 - Social interactions and collective user behaviors are relevant
 - Software-reliant
- SoS failures not typically due to single causes (or coding errors)

SYSTEMS OF **INDEPENDENT** SYSTEMS (SoISs)

Large system scale

- Failures are inevitable
 - Software at scale cannot be perfected
 - Recovery from failure must be part of the system design (and be verified)
- Can involve hundreds or thousands of components



Assurance

Assurance

- Justified confidence that a system will behave acceptably under all conditions of actual use
- How to know that system behavior is adequately constrained
 - so failures are not catastrophic



Assurance (for SoSs)

Assurance

- Justified confidence that a system will behave acceptably under all conditions of actual use
- How to know that system behavior is adequately constrained
 - so failures are not catastrophic

SoS V&V is more than testing

- V&V must involve analysis, modeling, and simulation
- Tests need to check the analyses, models, and simulations
 - Verify predictions
 - Uncover invalid assumptions
- Key question: what have you learned about untested cases when all tests run successfully?



Other SoS Assurance Considerations

Early lifecycle is not just about making requirements/design testable

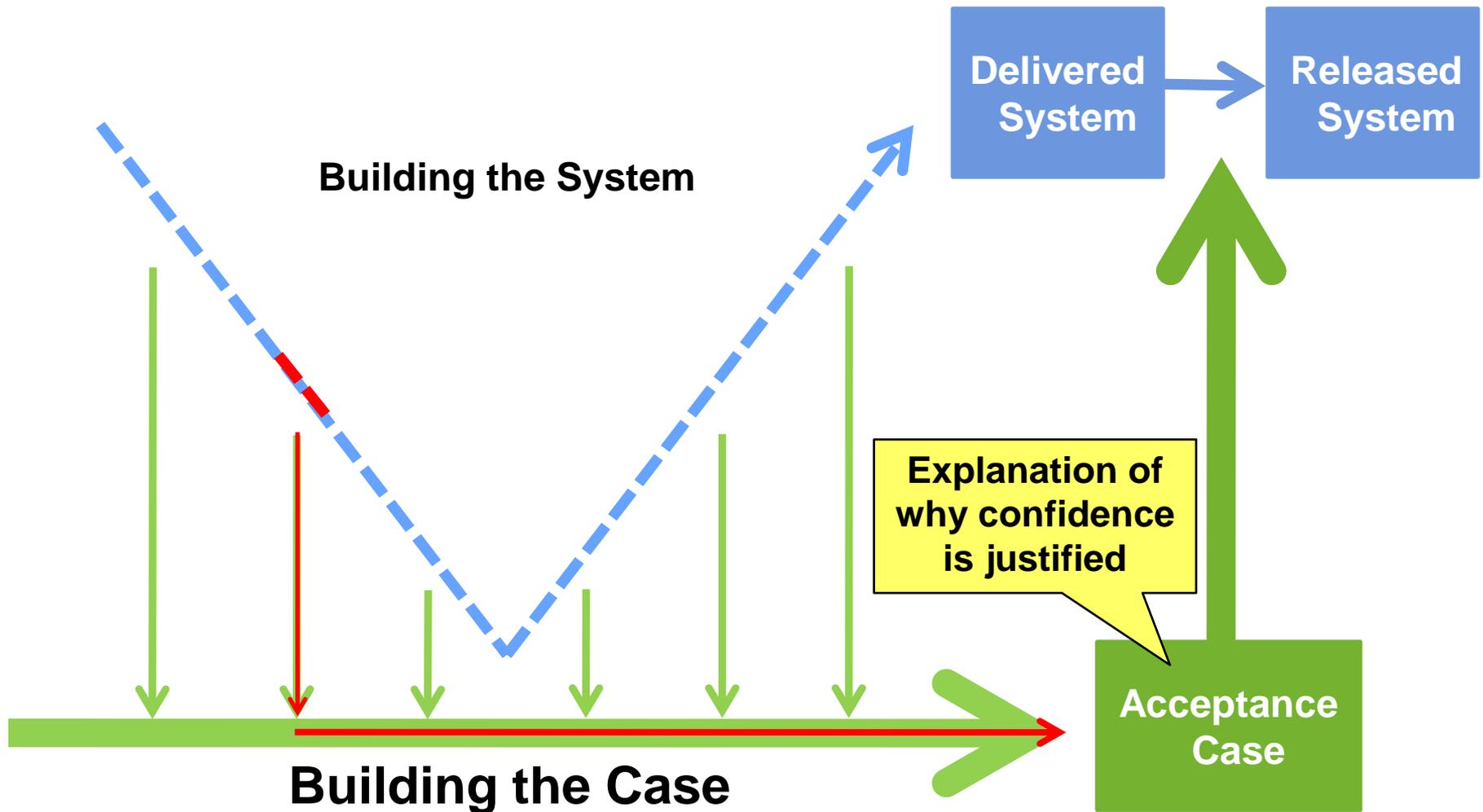
- Need to gather information showing that hazards to SoS operation have been recognized and dealt with appropriately
- Need to ensure that requirements and design do not introduce hazards

Continuous SoS evolution implies

- Continuous V&V and/or
- V&V focused on robustness against unanticipated changes/usage
 - Analysis
 - Run-time monitoring



Building Justified Confidence



Contact

John B. Goodenough

Fellow

Telephone: 412-268-6391

Email: jbg@sei.cmu.edu

U.S. Mail:

Software Engineering Institute

Carnegie Mellon University

4500 Fifth Avenue

Pittsburgh, PA 15213-3890



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.

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.

