Software at Scale

Systems of systems (SoSs)
- Constituents are decisionally autonomous, independently interacting entities
  - Socio-technical ecosystems
  - Software-reliant
- Constituents are developed (and evolve) independently
- SoS failures not typically due to single causes (or coding errors)

SYSTEMS OF INDEPENDENT SYSTEMS (SoISs)

My interest: SoS Software Assurance
- Justified confidence that a SoS will behave acceptably under all conditions of actual use
- How to know that SoS behavior is adequately constrained
  - so failures are not catastrophic
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 impact if miscoordination occurs? How is the impact detected, and mitigated?

Changed Assumptions

Social interactions and collective user behaviors are relevant

Failures are inevitable
- Software at scale cannot be perfected
- Coping with various types of failures (and failure effects) must be part of the system design
SoS Failure Modes

Tragedy of the commons
- Exhaustion of a common resource because there are no disincentives for overuse
  - SoS example: bandwidth

Unrecognized interdependencies
- Cascading failures (Italy power system)
- Interaction mediated by environment

Byzantine faults
- Different manifestations to different fault-free observers
- Not just replication inconsistencies

Failure to share info needed by another constituent

Inadequate enforcement of critical constraints (Leveson)
Inappropriate dependence on consistency (on some dimension)

Reliability/Robustness

For software at scale, reliability is not about coding errors
- It is about design (architectural) errors
- FMECA

Robustness: bounding effects of unanticipated/rare conditions

What constitutes SoS failure is not always obvious

Role of testing
- To validate models
  - Invalid assumptions
  - Predictions
- Provides evidence supporting conclusions about reliability/robustness
- Key question: what have you learned if all tests run successfully?

Incremental reliability re-evaluation methods
Needed Research

SoS failure modes and analysis methods for determining when SoS is sufficiently reliable/robust

Approach
- Create taxonomy of relevant SoS failure modes (with examples)
- Assess adequacy of mechanisms for gaining justified confidence that system is defended against relevant failure mode types
- Develop/adapt mechanisms where existing approaches are weak

Basis for success
- Can build on knowledge gained from distributed systems work
- Can look in different domains for examples

Impact
- Increased capability (due to increased confidence)
- Systems that behave well under stress and usage/env. changes

Contact

John B. Goodenough
Fellow
System of Systems Software Assurance
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.