Assurance Cases for Analysis of Complex System of Systems Software

Presented to the System of Systems Collaborators' Information Exchange
8 June 2010

Stephen Blanchette, Jr.

© 2010 Carnegie Mellon University

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.

© 2010 Carnegie Mellon University

Problem: SoS are Common, but Difficult

Huge systems of systems (SoS) development efforts

- Have become quite commonplace as aerospace/defense solutions
  - The Global Earth Observation System of Systems (GEOSS)
  - The US Army's Future Combat Systems
  - The US Coast Guard's Deepwater
  - Etc.
- Exhibit extreme complexity – they're hard to understand in detail
- Are difficult, even impossible, to test adequately using traditional methods
- *Use software as an enabler of SoS functionality*

Often, SoS elements are dispersed geographically, adding complexity and making predictability of behavior difficult

How can project management, stakeholders, and decision makers achieve some reasonable level of confidence that software for a large-scale, dispersed SoS will meet *operational needs*, even before development of much of the software?

© 2010 Carnegie Mellon University

Recent Experience Led to Assurance Case Use

An SEI team was tasked with answering this question in the face of several challenges:

- *the* software design was actually *many* software designs documented in many places and with a *tremendous* volume of data
- the designs were in different stages of completeness, but none would be complete until after many production decisions had to be made
- conclusions had to be based as much as possible on *actual data* rather than on optimistic plans and confident assertions

Constraints

- short time frame: a project level review had a hard deadline
- limited availability of personnel, all with varying levels of domain knowledge

© 2010 Carnegie Mellon University

Had to Relate Operational Needs to Software

Context

- "Deep dive" analyses were already underway in key technical areas

The top-level SoS requirements were expressed in military terms.

- A requirements analysis would be repeating work already done
- Errors in requirements traceability could skew the entire analysis

We decided against re-casting the operational needs in terms of software and instead analyzed the software contributions to the definitive characterization of those needs—the Key Performance Parameters (KPPs).

The size of the analysis space, the complexity of the task, and the desire to leverage data suggested an *assurance case* approach

- The analysis was a structured decomposition of each KPP into more precise statements that could be more readily assessed in terms of evidence.
  - Challenge: be logical and consistent but avoid accumulating too much detail
  - Use engineering judgment to leap from higher-level concepts to lower-level ones

© 2010 Carnegie Mellon University

An Assurance Case is a Structured Argument

An assurance case

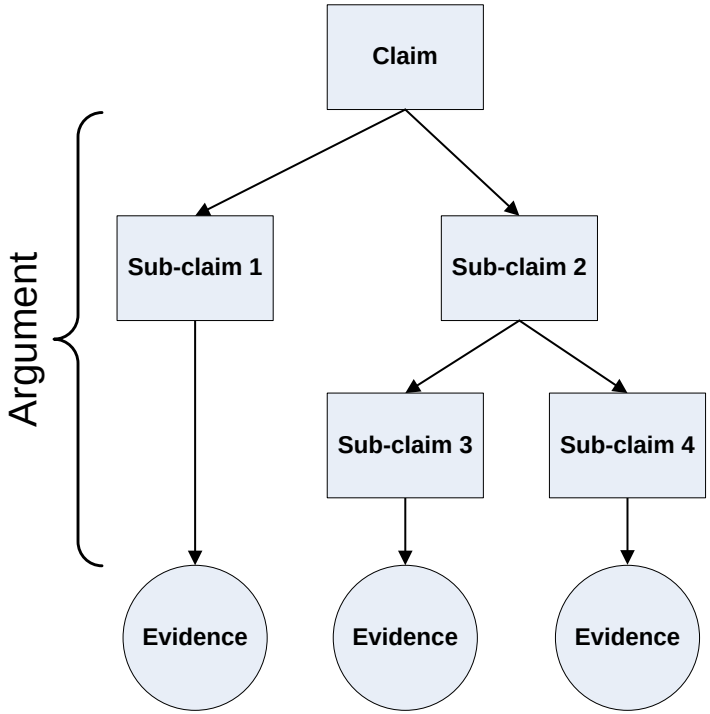
- is a generalization of a safety case
- presents an argument (similar to a legal case) that a system has or satisfies some property in a given context
- requires claims, evidence, and an argument linking evidence to claims
- should be sound and complete to justify belief in the main claim
- should be based on objective evidence

Goal Structuring Notation (GSN)

- graphically presents the *argument* by showing how *claims* are broken down into *sub-claims* until arriving at a sub-claim supported by *evidence*.

For our purposes, we used assurance cases to demonstrate that a SoS software design supported each of the SoS KPPs

- as in the claim, “The SoS will satisfy KPP k”



© 2010 Carnegie Mellon University

An Example

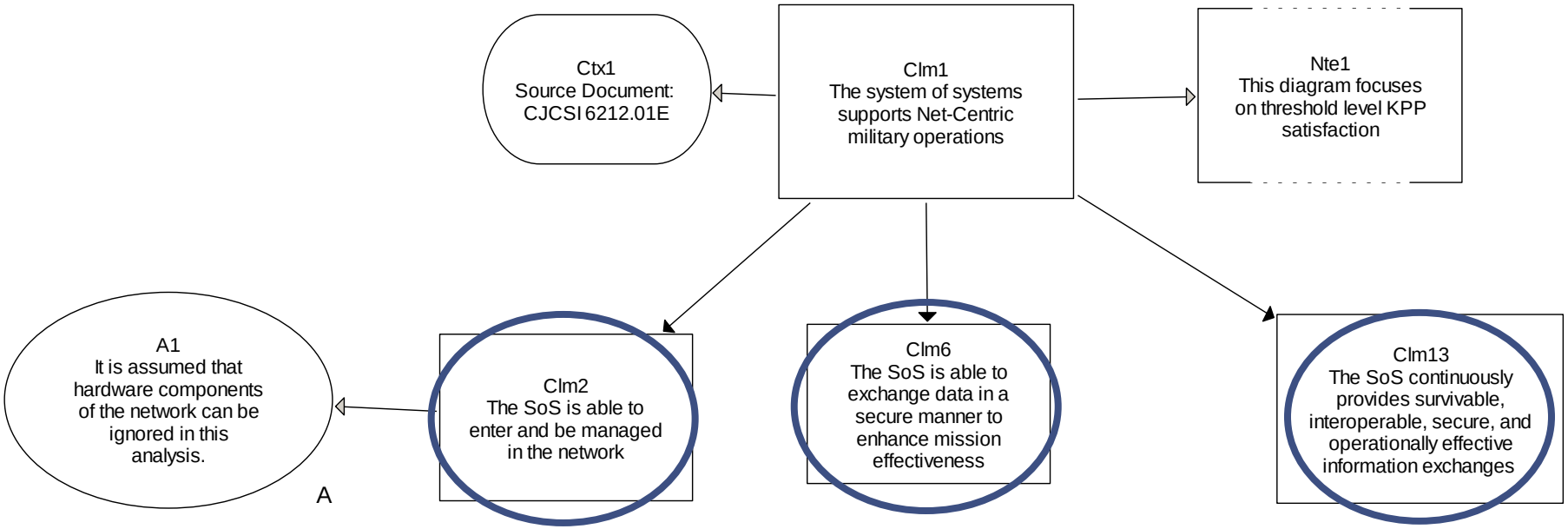
Assume we're developing a SoS for a DoD project

- The SoS must exchange information with other systems
  - Our SoS is subject to the Net Ready Key Performance Parameter (NR KPP):
    - **The system...must support Net-Centric military operations**. The…system…must be able to enter and be managed in the network, and exchange data in a secure manner to enhance mission effectiveness. The…system…must continuously provide survivable, interoperable, secure, and operationally effective information exchanges to enable a Net-Centric military capability.

Chairman of the Joint Chiefs of Staff Instruction, "Interoperability and Supportability of Information Technology and National Security Systems," CJCSI 6212.01E, 2008.

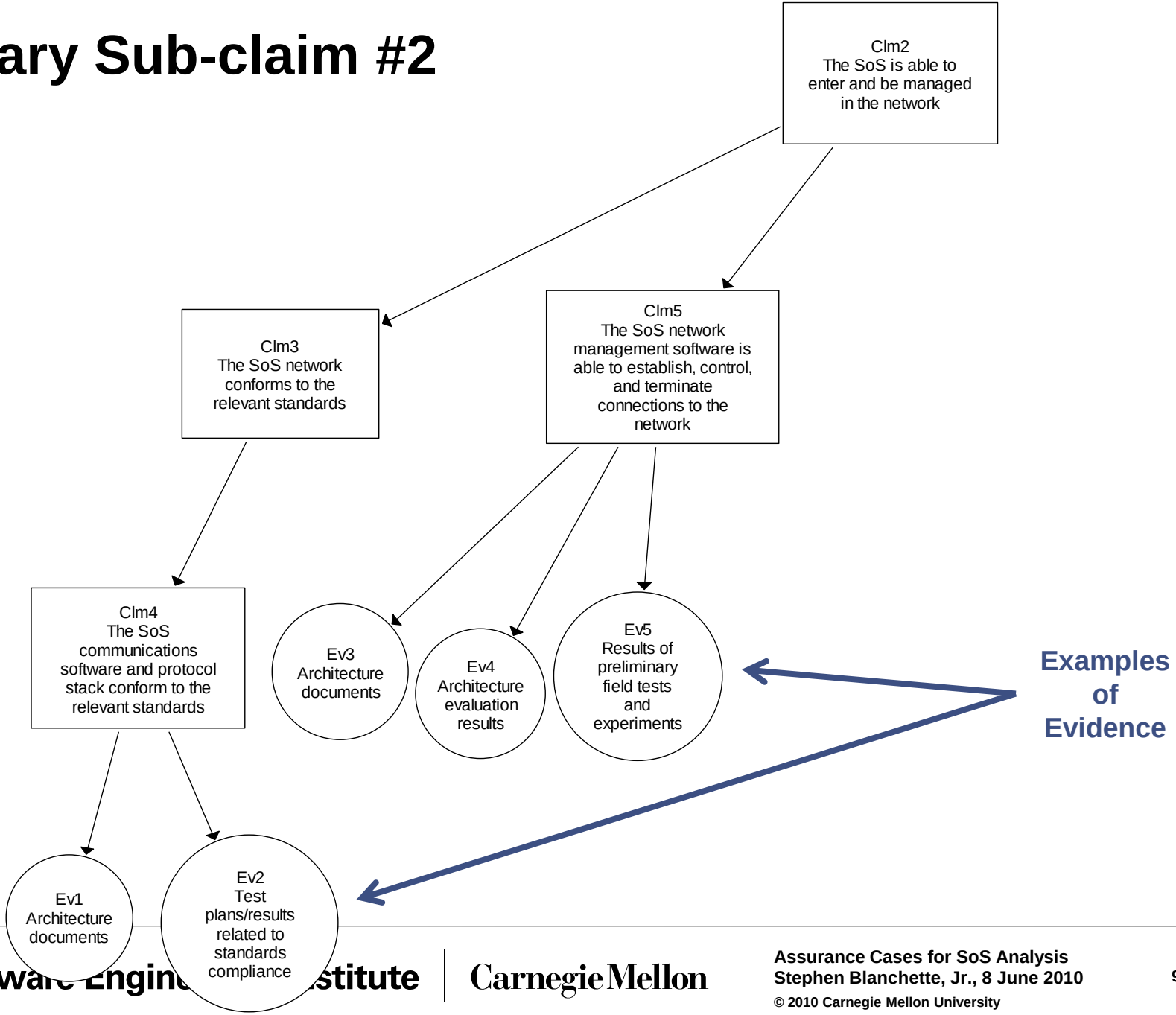
© 2010 Carnegie Mellon University

We Can Express the NR KPP Diagrammatically



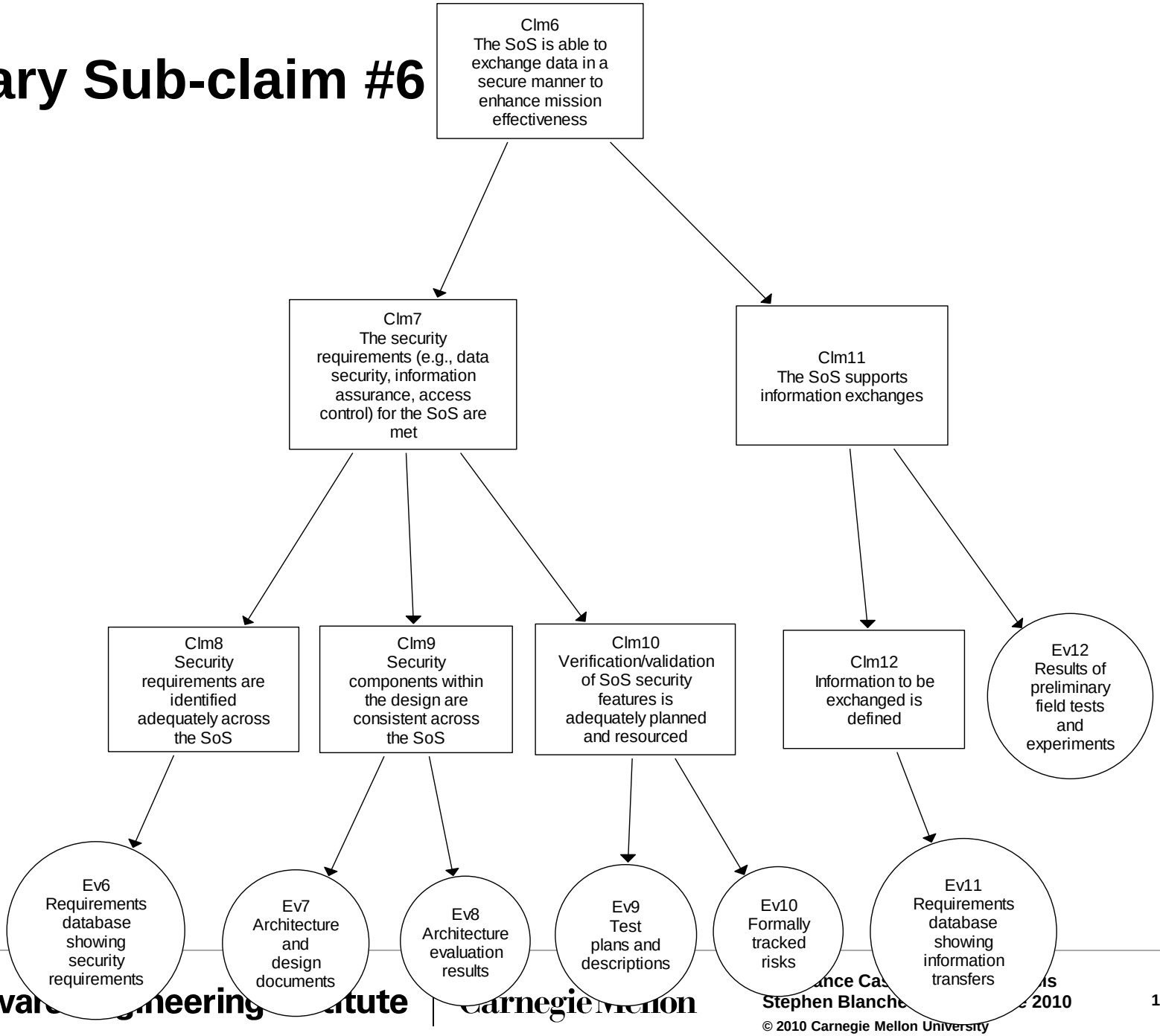
© 2010 Carnegie Mellon University

Primary Sub-claim #2



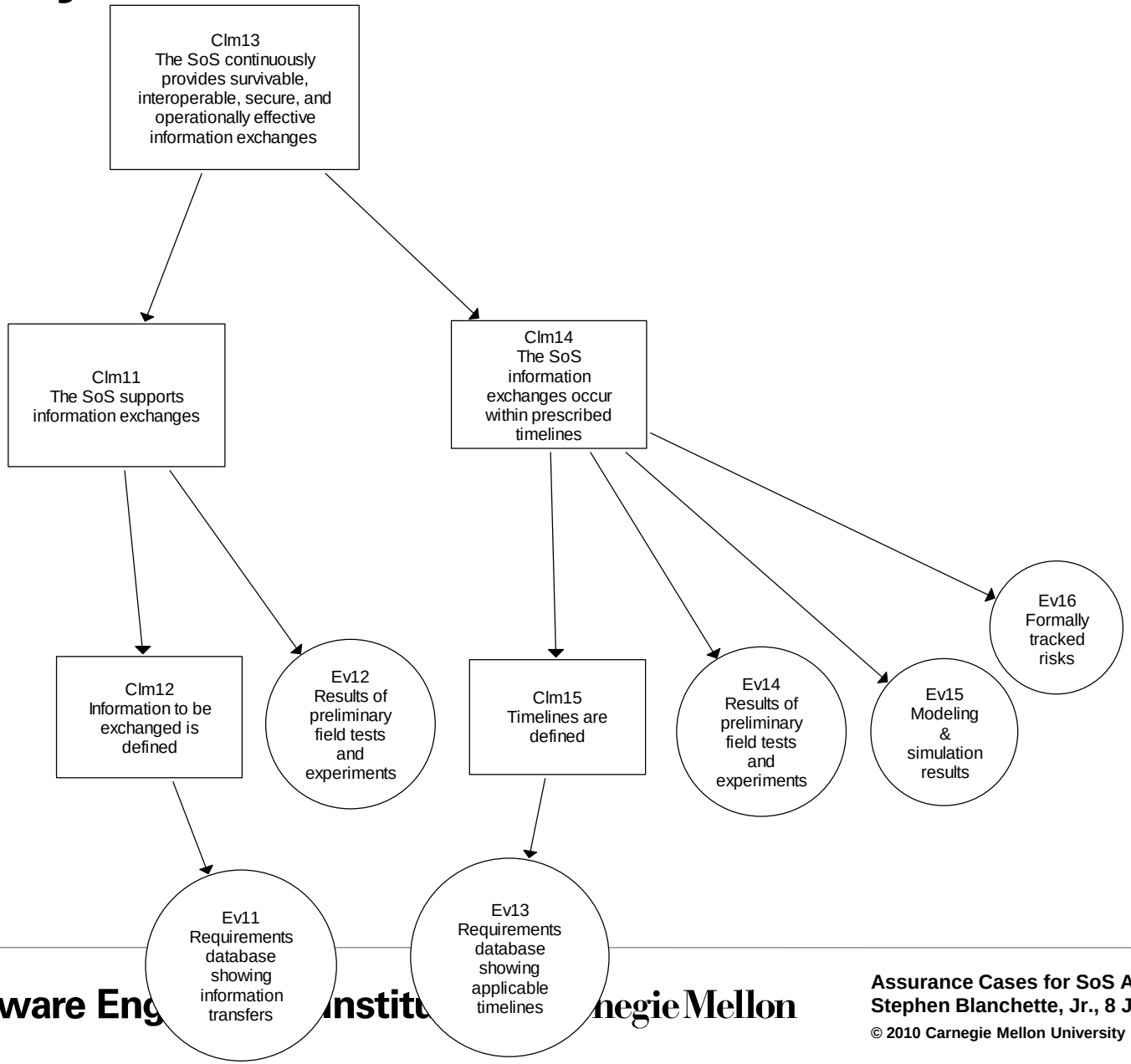
© 2010 Carnegie Mellon University

Primary Sub-claim #6



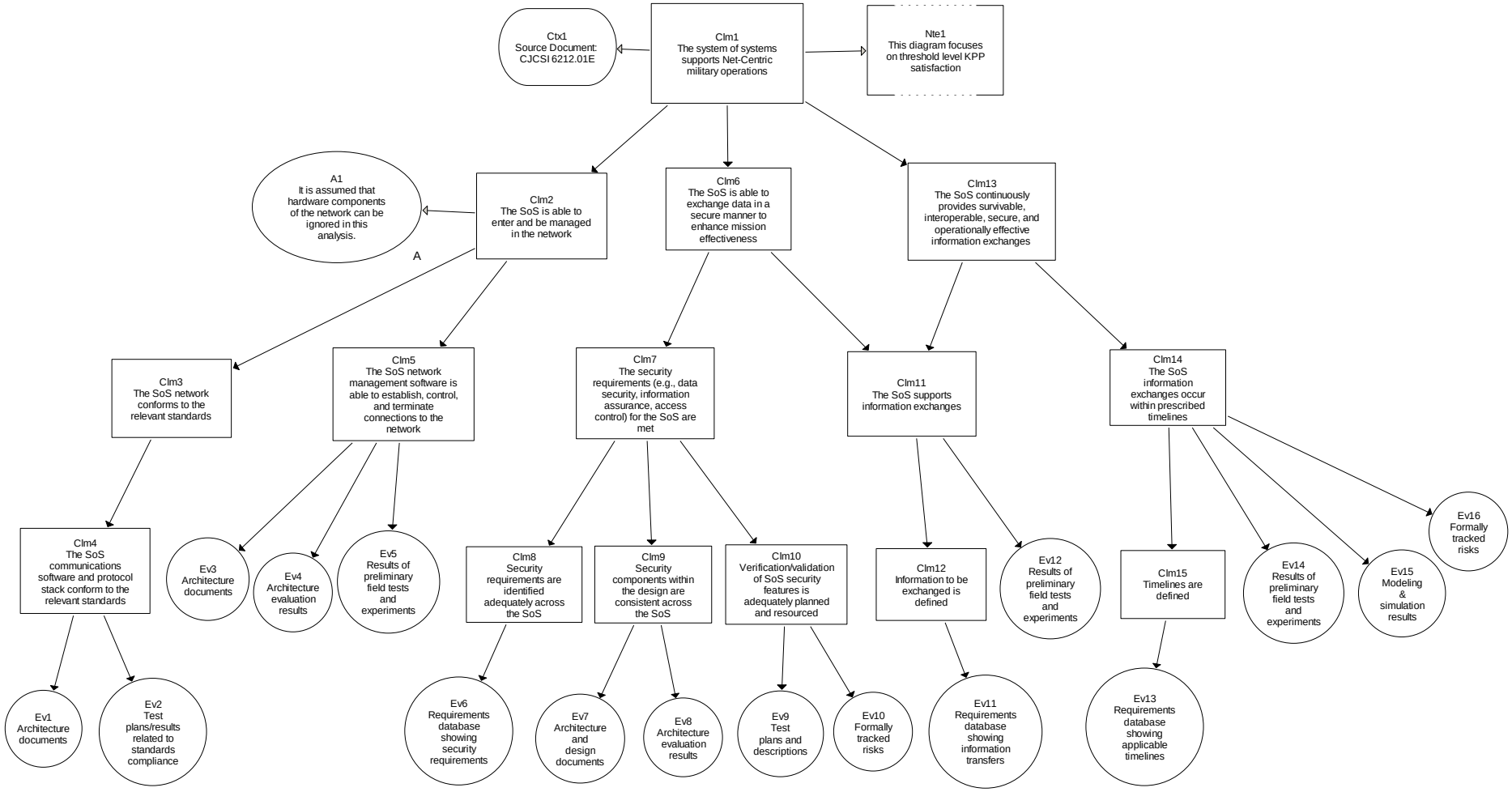
© 2010 Carnegie Mellon University

Primary Sub-claim #13



© 2010 Carnegie Mellon University

A Diagram Helps Visualize the Completed Case



Logic Should Hold…*Even Without the Diagram*

© 2010 Carnegie Mellon University

Scoring Can be Used to Express Risk

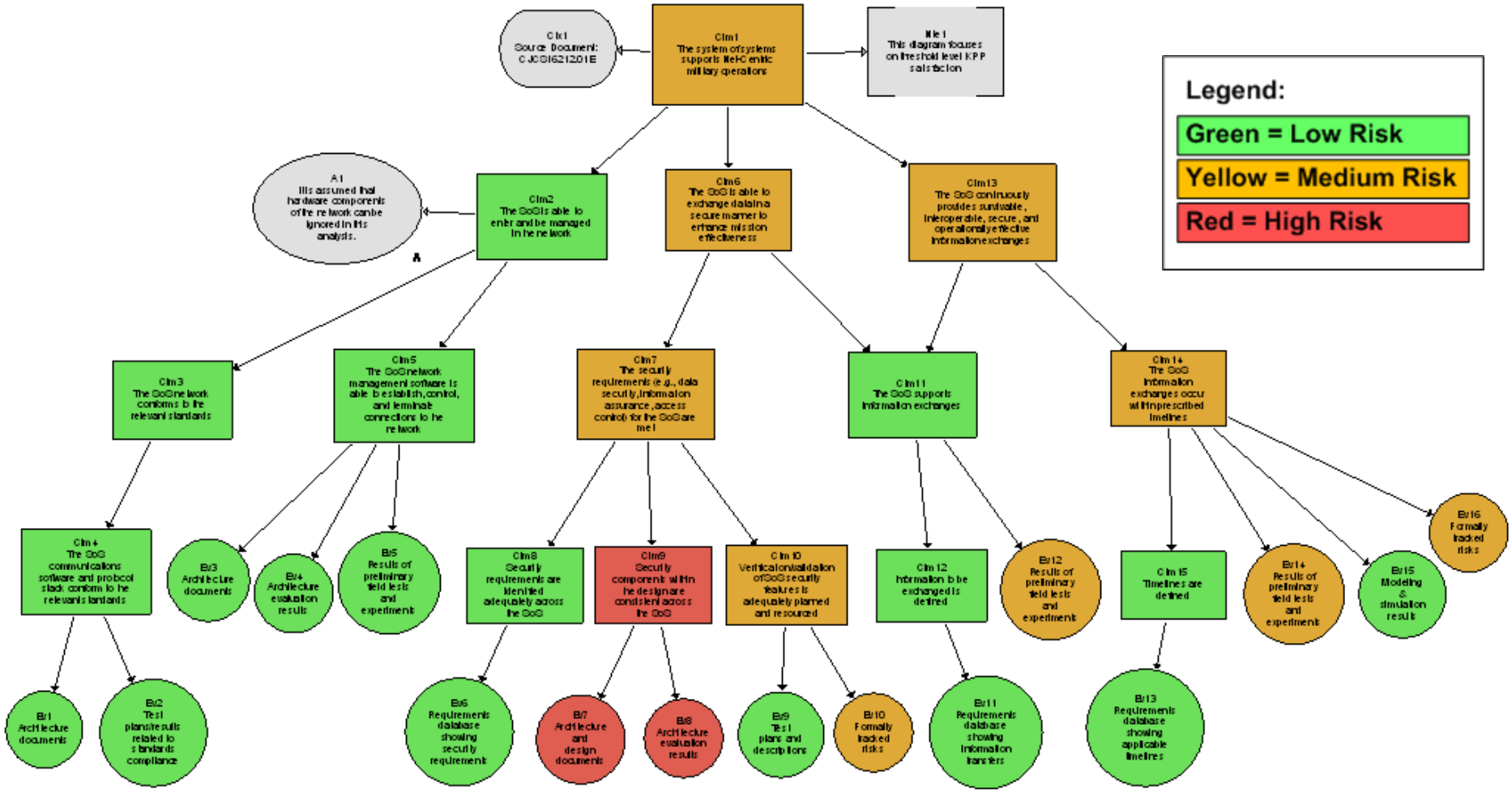
First, Develop Scoring Rules

For Evidence	
Green	Evidence is complete and adequate
Yellow	Evidence is incomplete or planned for the future
Red	Evidence is complete but inadequate, planned but now late, or non-existent
For Claims	
Green	All lower-level claims and supporting evidence are green
Yellow	Some lower-level claims and supporting evidence are a combination of yellow and red
Red	All, or an overwhelming majority of, lower-level claims and supporting evidence are red

Then, Work Upward from Evidence…

© 2010 Carnegie Mellon University

Scored Diagram Provides a Roadmap



Quality of the Evidence Drives Assessment of Claims…and Relative Risk

© 2010 Carnegie Mellon University

It Wasn't Really That Easy…

Up front planning is key

- Helps "socialize" the effort among affected areas of the project
- Minimizes need for repeated briefings
- Facilitates access to data & personnel

Software skills/domain knowledge are important

- Familiarity with technique and with operational concepts is needed, too

Starting with KPPs causes diagrams to grow quite busy very quickly

- Strong tendency to make very small steps in logic as one progresses down the analysis path in order to ensure the absolute soundness of the argument
- Engineering judgment needed to balance between precision in argument and comprehensibility

© 2010 Carnegie Mellon University

Assurance Cases are Helpful in the SoS Space

Assurance cases gave us a way of organizing a nebulous task and gave us a means of selecting among innumerable artifacts to study. They brought order to complexity.

- Due to time constraints we had to focus on big picture risks
- A more thorough analysis might have identified additional risks or strong points

The assurance case technique is a powerful tool for analyzing large and complex SoS software design.

- It provides a means of taking a crosscutting look at SoS
- It gives managers answers about design progress
  - rooted in facts and data instead of opinions based upon hope and best intentions
  - presented at a relevant level of detail to support decision making

© 2010 Carnegie Mellon University

Contact Information

Stephen Blanchette, Jr.

Deputy Chief Engineer, Army Programs

Acquisition Support Program

Telephone: +1 412-268-6275

Email: sblanche@sei.cmu.edu

U.S. mail:

Software Engineering Institute

4500 Fifth Avenue

Pittsburgh, PA 15213-2612

USA

Web:

www.sei.cmu.edu

Customer Relations

Email: info@sei.cmu.edu

SEI Phone: +1 412-268-5800

SEI Fax: +1 412-268-6257

© 2010 Carnegie Mellon University