NPS-AM-10-053

EXCERPT FROM THE PROCEEDINGS

OF THE

SEVENTH ANNUAL ACQUISITION RESEARCH SYMPOSIUM THURSDAY SESSIONS VOLUME II

Acquisition Research

Creating Synergy for Informed Change

May 12 - 13, 2010

Published: 30 April 2010

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **MAY 2010**	2. REPORT TYPE	3. DATES COVERED **00-00-2010 to 00-00-2010**

4. TITLE AND SUBTITLE **A Technique for Evaluating Complex System of Systems Designs**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Carnegie Mellon Software Engineering Institute,4500 Fifth Avenue,Pittsburgh,PA,15213**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
7th Annual Acquisition Research Symposium to be held May 12-13, 2010 in Monterey, California. U.S. Government or Federal Rights License

14. ABSTRACT
Complexity is the hallmark of most modern military systems. The desire to have legacy systems interoperate with new systems, and especially the mounting interest in developing ?systems of systems? (SoS) solutions, drive ever-more complexity into weapon systems. Complexity is further compounded by the increasing reliance on software to enable these systems. Existing forms of schedule- or event-driven reviews are inadequate to address the needs of software development in a complex SoS environment. What is needed is a true, evidence-driven, SoS-level evaluation capable of providing an overall assessment of, and insight into, the software development effort in that context. The Lifecycle Architecture anchor point is a technique used, typically, to evaluate the designs of single systems. This paper examines how, with some adaptation, the precepts of the Lifecycle Architecture anchor point can be scaled and applied to the system of systems domain.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**34**	

The research presented at the symposium was supported by the Acquisition Chair of the Graduate School of Business & Public Policy at the Naval Postgraduate School.

A Technique for Evaluating Complex System of Systems Designs

Stephen Blanchette Jr.—Stephen Blanchette, Jr. is the Deputy Chief Engineer for Army programs at the Carnegie Mellon Software Engineering Institute in Pittsburgh, Pennsylvania, where he specializes in acquisition improvement initiatives. He has over 23 years' experience in the defense industry as a software engineer and manager, including prior positions with United Defense, Stanford Telecommunications, and McDonnell Douglas. He is an associate fellow of the American Institute of Aeronautics and Astronautics and a senior member of the Institute of Electrical and Electronics Engineers. Mr. Blanchette earned a BS in Computer Science from Embry-Riddle Aeronautical University and an MA in Diplomacy from Norwich University.

Mr. Stephen Blanchette, Jr.
Acquisition Support Program
Software Engineering Institute
4500 Fifth Avenue
Pittsburgh, PA 15213 USA
412-268-6275 (voice)
412-268-5758 (fax)
sblanche@sei.cmu.edu

Steven Crosson—Steven Crosson is the Associate Director (Acting) for Software in the Army's Program Executive Office Integration. He has been with the Army for 6 years and the federal government for 7 years. He has held several project lead and chief engineer positions including the Combat Net Radio and United States Message Text Format software testing projects for the Communications-Electronics Life Cycle Management Command Software Engineering Center as well as being the chief software engineer for PM Future Combat Systems. Mr. Crosson earned a BS in Computer Engineering from the University of Delaware, and an MS in Software Engineering from Monmouth University.

Mr. Steven Crosson
United States Army – Program Executive Office Integration
6501 E. 11 Mile Road
SFAE-FCS-E/MS: 515
Warren, MI 48397 USA
steven.crosson@us.army.mil

Abstract

Complexity is the hallmark of most modern military systems. The desire to have legacy systems interoperate with new systems, and especially the mounting interest in developing "systems of systems" (SoS) solutions, drive ever-more complexity into weapon systems. Complexity is further compounded by the increasing reliance on software to enable these systems. Existing forms of schedule- or event-driven reviews are inadequate to address the needs of software development in a complex SoS environment. What is needed is a true, evidence-driven, SoS-level evaluation capable of providing an overall assessment of, and insight into, the software development effort in that context. The Lifecycle Architecture anchor point is a technique used, typically, to evaluate the designs of single systems. This paper examines how, with some adaptation, the precepts of the Lifecycle Architecture anchor point can be scaled and applied to the system of systems domain.

Keywords: System of Systems, SoS, LCA, complexity

Acknowledgements

The authors gratefully acknowledge the contributions of Dr. Barry Boehm of the University of Southern California, who was instrumental in developing the technique described in this paper.

Introduction

Modern military systems are complex undertakings. The desire to have legacy systems interoperate with new systems, and especially the mounting interest in developing “systems of systems” (SoS) solutions, drive ever-more complexity. The problem is exacerbated by the reliance upon software to accomplish much of the underlying functionality; indeed, software often is the integrating element of an SoS. This complexity challenges the ability of engineers, managers, and users to achieve a solid understanding of an SoS during its development. Frequently, the SoS stakeholders (apart from the software developers themselves) have a very limited understanding of software or its contribution to the overall SoS. Yet, understanding an SoS during development is crucial to managing the development effort efficiently and to delivering quality products to the warfighter within schedule and budget.

Traditionally, schedule- or event-driven reviews have been a crucial element of most major development projects. From a software perspective, such reviews tend to focus on different aspects of development: producibility reviews focus on the ability to produce the software within available resources; capability reviews focus on the services being provided by the software; integration and test reviews focus on the readiness of the software to enter or transition between those phases of the development life cycle; schedule reviews focus on the development effort’s adherence to planned timelines; and so on. Although these different types of reviews provide valuable insights into the software development project at various stages of the lifecycle, the sum of these reviews is not sufficient to address the needs of software development in a complex SoS environment.

Most overall-systems reviews concentrate mainly on the functional definitions of system artifacts, and often treat evidence that the artifacts described will meet the system’s key performance parameter requirements comparatively lightly. Further, in order to support the needs of manufacturing planning and long-lead purchasing, such reviews tend to occur at points in the system development lifecycle ahead of the majority of the software development work. The resulting gap between maturity of SoS constituent systems and maturity of SoS software leads to uncertainty about the sufficiency of the overall SoS solution at a time when commitments must be made. Thus, traditional review approaches simply are inadequate when used as a means for understanding and evaluating software at an SoS level. How can stakeholders have confidence in an SoS that relies heavily upon software, when production commitments must be made well before completion of that software?

Prior to its 2009 cancellation, the US Army’s Future Combat Systems program faced just such a dilemma. Needed was a true SoS-level evaluation capable of providing an overall assessment of, and insight into, the software development effort in that context—one that could provide a review/assessment of how the developed software capability enabled the program’s required operational capability. As a solution, the program hypothesized that something like a Lifecycle Architecture anchor point review (often referred to simply as LCA), conducted at the SoS level, could answer the question. Originally a software development notion in the Rational Unified Process (RUP), the LCA marks the conclusion of the elaboration phase of software development, when the requirements baseline is set and the architecture is complete (Kroll & Kruchten, 2003; Boehm, 1996). As extended to systems engineering in works by Richard Pew and Anne Mavor (2007) and Barry Boehm and Jo Ann Lane (2007), the goal of anchoring events such as the LCA is to assess program risk by examining evidence provided by the developers that a system developed to the architecture can satisfy the requirements. The LCA, in particular, strives to ensure that a system, as architected, can be constructed while also meeting planned cost and schedule targets.

This paper reports on the extension of the Lifecycle Architecture anchor point concept to the SoS level and describes its application to the software and computing elements of the former Future Combat Systems program.

The Lifecycle Architecture Anchor Point

Nominally, the LCA anchor point occurs at a stage in the development lifecycle where stakeholders evaluate the work that has been completed through the elaboration phase (the phase in which requirements and architecture models are largely complete) and assess the risks of moving forward into the construction and transition phases (the phases in which design, implementation, and testing/validation are performed). The name LCA derives from the nature of the review: it is a look-ahead through the lifecycle, conducted at a point where the architecture is sufficiently mature to allow reasoning about the relative risks of continuing to the construction phase of development. It is a foundational event, one where project stakeholders come together and agree to move forward, hence the term “anchor point”. The LCA differs from traditional milestone reviews such as preliminary design reviews (PDRs) and critical design reviews (CDRs), which tend to focus superficially on voluminous system description data, in that the LCA is a risk-based assessment focused on the feasibility of proceeding with work.

Central to the LCA is the notion of feasibility rationale, which documents evidence, provided by the developers, that the proposed architecture can be implemented to satisfy its requirements within the defined schedule and project budget. To justify the confidence of all stakeholders in moving forward into the latter phases of the development life cycle, the evidence must be both objective and internally consistent. The LCA is not just a technical assessment, but also a programmatic one. Successful completion of the LCA anchor point represents a commitment by all stakeholders to proceed with the program, based on objective evidence that the risks of doing so have been identified and sufficiently mitigated to provide a reasonable chance of project success. In contrast, insufficient or highly subjective evidence will not engender the confidence among stakeholders needed for them to commit to further development. Importantly, risks are addressed at a time when they are handled more easily and inexpensively than if they are allowed to propagate to later development stages.

The LCA feasibility rationale is relatively straightforward in the context of a traditional system development project. For a complex system of systems program, the feasibility package becomes less clear. Simply rolling up results from the LCA anchor points of constituent systems, for example, is not a sufficient approach because important inter-system and SoS implications might easily be missed. The feasibility of executing each individual software package/system does not necessarily correlate to the feasibility of executing the entire SoS.

Extending the LCA Process For The FCS SoS

Prior to its cancellation, the Future Combat Systems (FCS) program extended the notion of a Lifecycle Architecture anchor point to the SoS level as a means of evaluating on-going software development activities across the lifecycle in terms of risk with respect to meeting the program's Key Performance Parameters (KPPs).

The first step was to set up a team of experts from the Software Engineering Institute, the University of Southern California, and the Fraunhofer Center for Experimental Software Engineering at the University of Maryland (hereinafter referred to as the *SoS LCA Team*), with representatives from the Army providing guidance and representatives from the program's Lead System Integrator (LSI) facilitating access to program artifacts and personnel. Team members brought a range of programmatic and technical expertise to the effort, including a deep understanding of the usual LCA process for software. They also brought the degree of independence necessary to assure both the LSI and the Army of an unbiased result.

The SoS LCA Concept

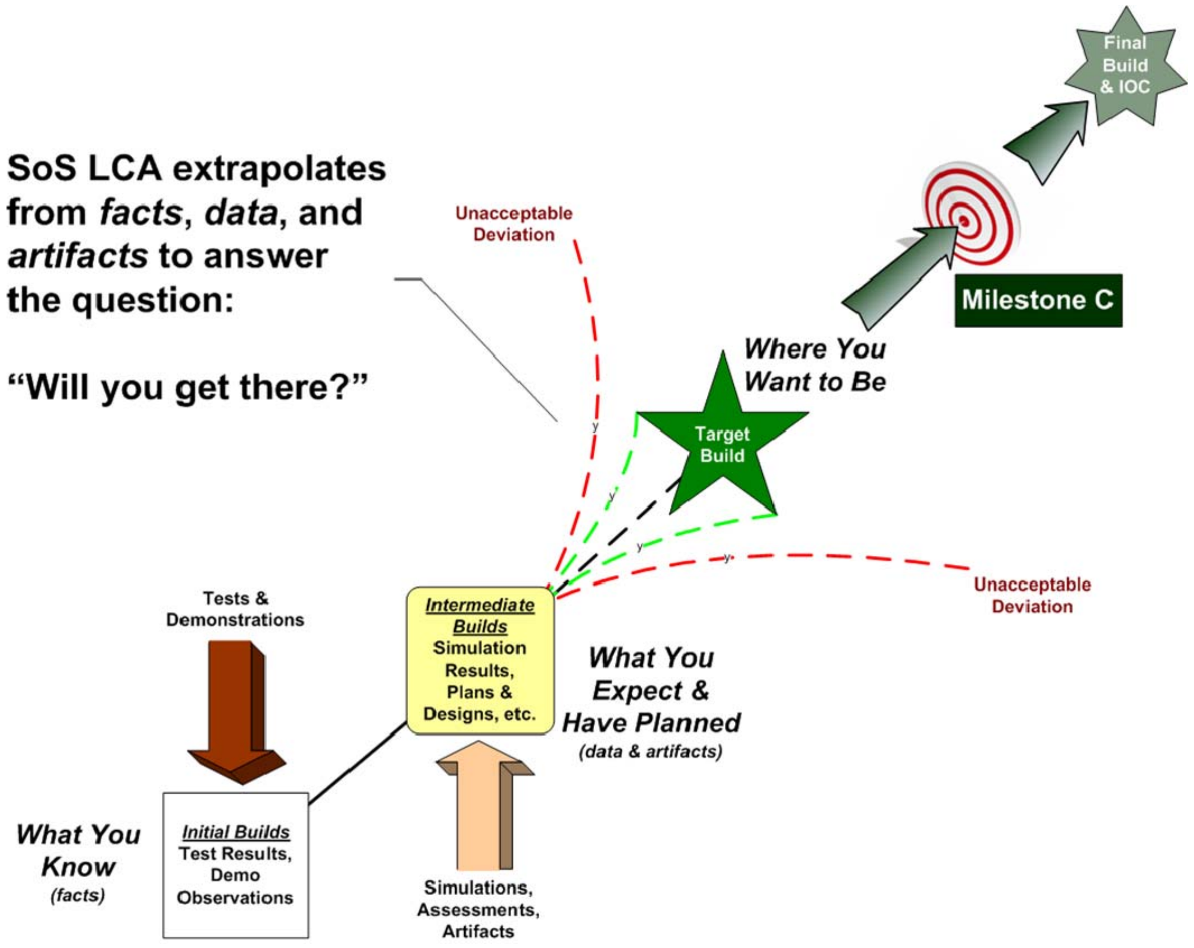
In defining the SoS LCA review process, the SoS LCA Team followed a few key guidelines set by program management:

- Answer the following questions:
  - Can what is planned to be built really be built?
  - Will the various pieces add up to the desired whole?
  - Are the risks clearly identified and mitigated?
- Base answers on evidence rather than unsupported assertions.
- Discover issues early so that they can be fixed sooner rather than later.
- Build the confidence needed in the software/computing system development plans for a successful Milestone C decision for the overall program.

Rather than finding fault and assigning blame, the goal of the SoS LCA was to surface problems as early as possible so that they could be fixed at minimum cost. Equally important for FCS was the building of confidence in the software and computing system development plans, since an entire build of software would not yet have been developed at the time of the program's planned Milestone C review.

The FCS program had been executing traditional LCA reviews at the supplier and first-tier integration levels; however, simply rolling-up the results of these lower level reviews would have invited missing critical subtleties in the cross-system relationships that are the essence of an SoS. Also, the lower level LCA reviews focused narrowly on specific builds, often at different points in time. The SoS-level LCA was required to assess feasibility across future builds. Further, while other FCS software reviews focused on the plans for the immediately upcoming phase of a given build, the SoS LCA had to consider existing data and results from prior builds as well. The SoS LCA Team had to construct an SoS LCA evaluation process that had as its foundation the best-available facts and data as the basis for projecting forward.

As shown in Figure 1, the FCS SoS LCA process extrapolated, from what was known through what was expected and planned, the likelihood of reaching the desired endpoint within a tolerable level of risk. In the case of FCS, the desired endpoint was an acceptable implementation of software Build 3 Final (B3F) to support a successful Milestone C decision (consequently, the SoS LCA results were a key feeder into the program's SoS PDR).


SoS LCA extrapolates from *facts*, *data*, and *artifacts* to answer the question:
"Will you get there?"
Final Build & IOC
Unacceptable Deviation
Milestone C
Where You Want to Be
Target Build
Unacceptable Deviation
Tests & Demonstrations
Intermediate Builds
Simulation Results, Plans & Designs, etc.
What You Expect & Have Planned
(data & artifacts)
What You Know
(facts)
Initial Builds
Test Results, Demo Observations
Simulations, Assessments, Artifacts

The SoS LCA Assesses the Likelihood of Achieving an Acceptable Outcome

Without waiting for tests on the final implementation, the SoS LCA sought to determine if FCS software could be built and if it would satisfy program needs. The basis of the extrapolation was objective evidence: *facts* and *data* in addition to other artifacts. Facts took the form of test results and demonstration outcomes for software that already had been built. Data consisted of simulation results that predicted operational performance of planned designs as well as the results of various technical assessments. The remaining artifacts of interest included the various plans, designs, schedules, and so on, that formed the basis of the work yet to be done. Engineering work products were evaluated for completeness and adequacy, as well as for consistency with other work products and plans. The SoS LCA focused on test data and results (as well as software producibility analysis) to evaluate the current capability of FCS software and to project the ability to meet program needs. The goal was to determine if development was on an acceptable trajectory or if deviations of sufficient significance required adjustments to development planning or execution.

Scope

On a very large SoS program such as FCS, there is no practical way to review all of the data one might wish to analyze in order to draw conclusions. Such a detailed effort might require thousands of labor hours spread over many months. While the cost alone would be a significant deterrent, the time factor also would be of concern. The longer it takes to complete the examination of all the evidence, the higher the risk that analyses performed early in the process will be invalidated by ongoing development work (due to prior problems or unknowns being resolved or new issues being introduced).

On FCS, program management had decided to limit the scope of the SoS LCA to the software and computer-processing elements of the program, but even that limitation left the range of investigation too broad. It quickly became apparent to the SoS LCA Team that executing a traditional LCA at the SoS level, even one constrained to the software and computing system elements of the program, would not be feasible. Merely coordinating an effort of that magnitude would become a project unto itself. Such an LCA would be too large, too complex, and too resource-intensive to be managed successfully. Instead, the team had to determine what a modest-sized team with limited resources could reasonably accomplish. The result was a focus on high-payoff areas for the program. These focus areas were not necessarily risk areas, but rather crucial areas to successful development of the FCS software:

1. ability to meet schedules within and across increments
1. ability to meet budgets within and across increments
2. ability to integrate across Integrated Product Teams (IPTs) and suppliers (including adequacy of requirements, architectural consistency, and so on)
3. ability to achieve/maintain interoperability among independently evolving systems (including Current Force platforms)
4. ability to coordinate multiple baselines for the core program, spin outs, and experiments
5. ability to manage co-dependencies between models, simulations, and actual systems
6. feasibility of meeting required performance levels in areas such as safety and security
7. maturity of technology considering scalability requirements
8. supportability of software
9. adequacy of commercial-off-the-shelf (COTS) evaluations and mitigations

A review of the results from lower level LCAs showed that these focus areas were commonly cited as problems, thus validating the list. With the scope bounded, the SoS LCA Team next worked on developing a process model for the SoS LCA.

An SoS LCA Process Model—First Cut

Figure 2 shows the initial SoS LCA process model. As envisioned, several analysis teams, consisting of the experts from the SoS LCA Team as well as representatives from the government, LSI, and the program’s suppliers (referred to as One Team Partners, or

OTPs), were to be formed. Each team would tackle a single focus area, examining the relevant facts, data, evidence, and plans and developing conclusions based on those examinations. The results from each analysis team would be collected, summarized, and reported on by the SoS LCA Team, with a final report that detailed findings and recommendations serving as input to the program's SoS PDR. The SoS LCA final report, combined with results from the SoS PDR, would provide the basis for management decision making regarding any corrective actions.

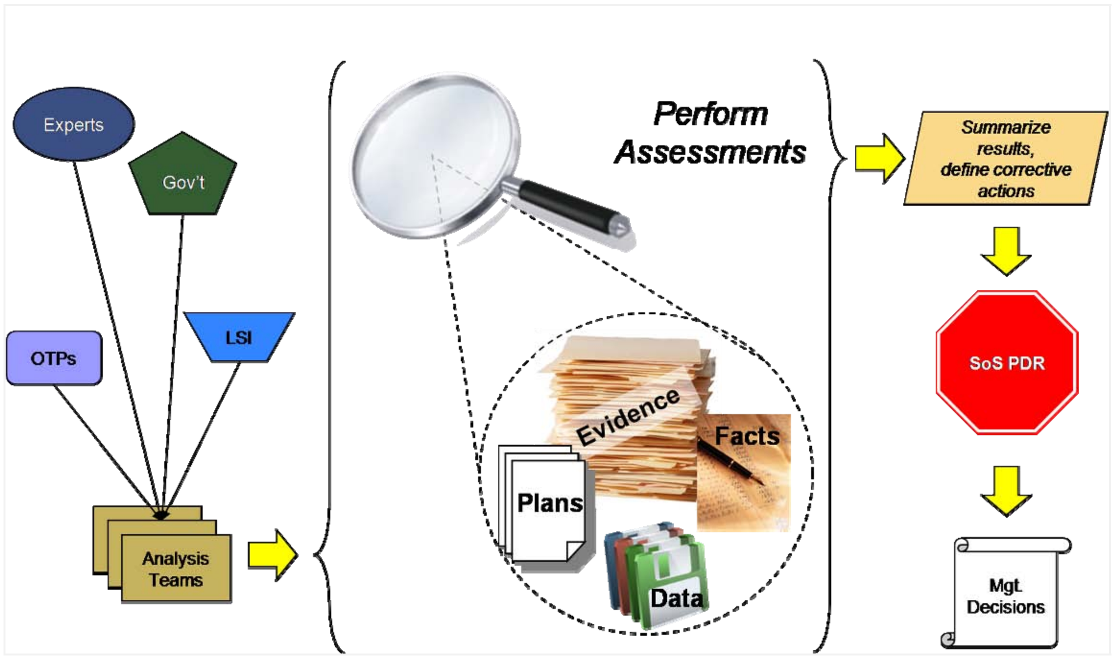


Figure 1. The Initial SoS LCA Process Model

Although the process model seemed to be straightforward, team members' experiences indicated that piloting the SoS LCA process on a small scale would be a wise first step. A pilot would allow the team to figure out the details of executing the process and discover any invalid assumptions and process inefficiencies.

The SoS LCA Pilot

Key considerations in planning for the SoS LCA pilot effort were its scoping and timing. It was important to limit the scope of the pilot to be executable relatively quickly, while at the same keep it broad enough to be useful as a test run for the larger effort. Equally challenging was scheduling the pilot. On a program with so many separate but interacting aspects, it seemed as if there was always a conflicting event to draw time, attention, and resources away from the SoS LCA pilot. As the pilot was not part of the original program plan, the need to minimize disruption to ongoing program activities served as yet another complication.

As it happened, the lead of the C4ISR[1] team volunteered his area of responsibility to participate in the SoS LCA pilot because he saw value in having some independent review

[1] C4ISR stands for command, control, communications, computers, intelligence, surveillance, and reconnaissance.

of the work being performed by his team. The C4ISR area, however, was quite sizeable with many interdependent elements, which was good from the perspective of piloting the process but challenging in terms of completing the pilot in a short timeframe. The SoS LCA Team negotiated with the C4ISR team lead to adjust the pilot scope to be achievable yet still worthwhile. The negotiated scope included a subset of the 10 SoS LCA focus areas, coupled with the C4ISR lead's principle areas of concern. The timing of the pilot also aligned well with a previously planned review cycle for the C4ISR team.

After first socializing the notion of the pilot among the C4ISR team members so that expectations on both sides were clear, the SoS LCA Team next set about gathering data. Team members divided the focus areas among themselves based on their respective backgrounds and expertise. The Team spent about three weeks gathering evidence from existing C4ISR plans and technical documents; no new documents were to be created for the purposes of the pilot.

Almost immediately, the pilot began serving its purpose. One early discovery was that SoS LCA Team members were not in full agreement about the goals or method of conduct of the pilot. Thus, the pilot was an invaluable experience both from the perspective of checking and refining the process and from the point of view of developing a common mission understanding within the SoS LCA Team.

Another revelation was the considerable amount of misunderstanding between the SoS LCA Team, the C4ISR team, and its various performer organizations (despite efforts to set clear expectations at the outset). Documentation and evidence furnished by the C4ISR team generated more questions than answers, requiring numerous follow-up teleconferences and requests for additional documents. The documents provided often had timestamps differing by several months and frequently made what appeared to be incompatible assumptions. Trying to resolve these inconsistencies without interrupting the performers' work gave the SoS LCA Team an appreciation of the difficulty to be expected when performing a complete assessment.

The SoS LCA Team found little documentation to support an independent conclusion that the C4ISR plans and approach were feasible. This outcome was not because the C4ISR team had not done its job; on the contrary, there was ample evidence of significant engineering effort. However, there was no contractual obligation for suppliers to demonstrate the feasibility of their work, per se, outside of customary test and validation methods, which placed a heavier burden on the analysis team in trying to understand how completed work related to future work. Since the crux of the LCA is reliance upon evidence rather than verbal accounts and assurances, the SoS LCA Team realized that the full assessment would require extra time for data gathering and a deeper probe of the available evidence in order to ensure the veracity of findings.

Despite the noted difficulties, the pilot generally was a success in that it helped clarify the SoS LCA Team's thinking about conducting the full-scale SoS LCA. It demonstrated that program personnel would be too busy to participate significantly in gathering and examining feasibility evidence, which would necessitate a larger effort on the part of the analysis team. It also demonstrated the need to set expectations among developers and management continually to ensure their ongoing support of the process. The pilot also highlighted the need to review emerging findings with stakeholders to ensure accuracy. Meanwhile, the C4ISR Team also benefitted from the pilot by being able to raise issues that had been of concern to them for management attention, with the added weight of independent analysis to support them.

Some Process Refinements

As a result of the SoS LCA pilot and other program activities, the SoS LCA Team made some adjustments both to the focus areas and to the process model.

First, in consultation with program management, the SoS LCA Team modified the focus areas. The initial list proved to be too vague and overly oriented toward programmatic (e.g., non-technical) considerations. The modified list represented the essential FCS capabilities, the "non-disposable" elements of the program. These elements (see Table 1) were ones that had to work in order to ensure that the program could produce a viable SoS capability for the Warfighter.

Table 1. Final SoS LCA Focus Areas

Focus Area	Description
Fusion and Data Distribution	Includes: *Distributed Information Management*–effectiveness of data communications among platforms and systems over a network, analysis of data reliability, latency, etc. *Distributed Fusion Management*–effectiveness of utilizing several sources to gather and distribute fusion data, including network usage, data reliability, latency, etc. *Rapid Battlespace Deconfliction*–effectiveness of providing, recognizing, and utilizing battlespace deconfliction data passed among Brigade Combat Team (BCT) platforms
Quality of Service (QoS)	Analysis of message prioritization effectiveness, delivery rates, methods of prioritization, etc.
Information Assurance (IA) & Security	Review of the status of efforts to develop and test FCS IA components, and a determination of the attainment of necessary functional capabilities
Software Performance	Analysis of the projected ability of fielded FCS software (e.g., algorithms) to enable planned capabilities in a comprehensive, end-to-end operational environment
Display Management and Task Automation/Task Integration Networks (TINs)	Analysis of the performance, usability, and automation capabilities demonstrated in Warfighter Machine Interface (WMI) displays
Key Battle Command Systems	Includes: *Network Management System (NMS)*–review of current status of development of the Network Management System (NMS) system, as well as analysis of the current state of general network management issues *Integrated Computer System (ICS)*–assessment of the development and issues related to the various versions of the ICS system. Determination of the ICS' ability to meet FCS design needs *Centralized Controller (CC)*–analysis of the current state of the CC and its ability to meet necessary functional capabilities

The results of investigations into these focus areas formed the detailed technical basis for the final report to management.

Second, the realities of personnel availability necessitated slight adjustments to the process model. Figure 3 shows the changes, which were encapsulated entirely within analysis team formation (i.e., inside the red oval denoted by the gray arrow on the left-hand side of the diagram).

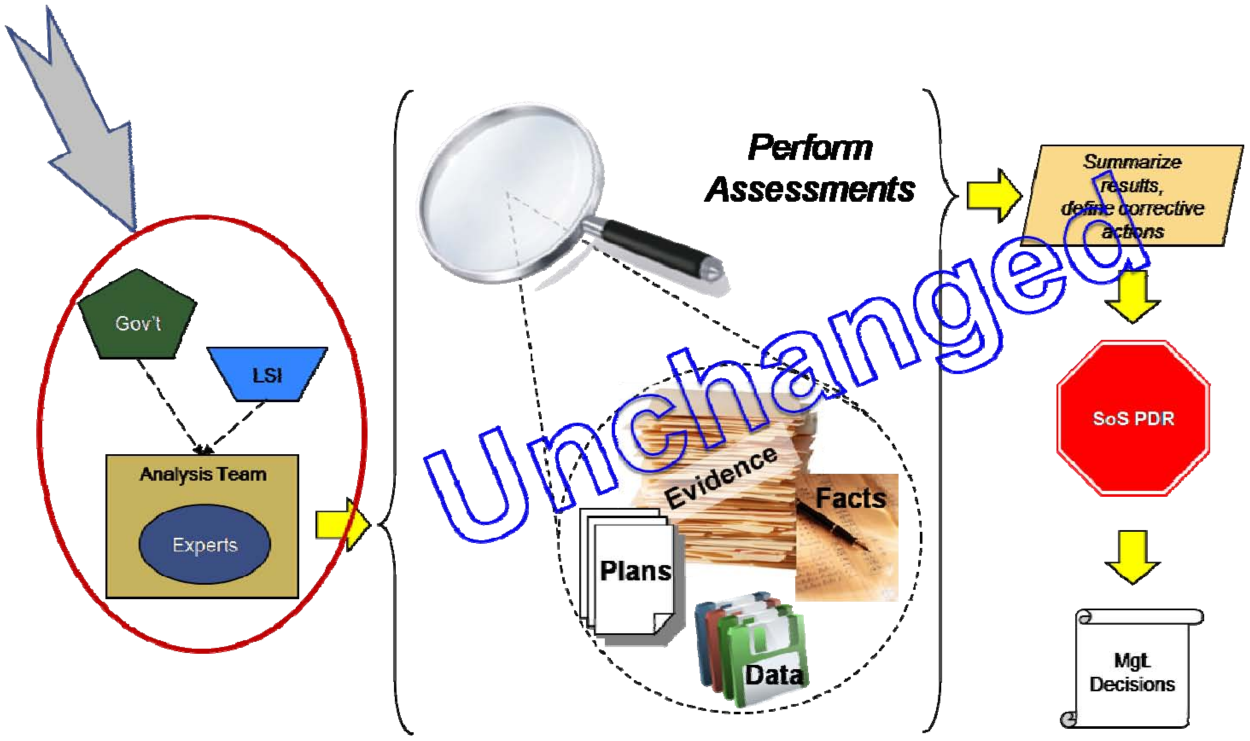


Figure 2. SoS LCA Process Model Showing Refinements

As shown in Figure 3, instead of several analysis teams, there was only one, which was composed entirely of technical experts from the SoS LCA Team, although Army representatives provided guidance and an LSI representative facilitated access to documentation. Suppliers were unable to participate in the analysis process at all, although they were called upon to provide data and, occasionally, clarification of the data. The SoS LCA Team divided the focus areas among themselves; in each area, typically one or two team members had responsibility for identifying the types of artifacts needed for evaluation, reviewing them, and rendering recommendations based on review findings. The entire analysis team reviewed results as they emerged and briefed status to program management at frequent intervals.

Capstone Analyses: Tying It All Together

While detailed technical analyses are the foundation of the SoS LCA, they are inadequate for communicating status and risk to stakeholders who may have limited expertise in those technical areas. In particular, the highest levels of program management need to be able to relate the SoS LCA findings to overall program outcomes in order to make informed decisions about risk mitigation. Further, the thread to tie the focus areas together in a meaningful way was missing. Needed was a means of looking at the results in a holistic manner and interpreting them in way that would be relevant to management decision-making. The solution involved two capstone analysis efforts, one from a technical perspective and the other from a cost and schedule perspective. The end-state design analysis summed up the findings of the technical focus areas in a way that made the technical findings relevant for management decision-making. The software producibility analysis looked at the feasibility of developing the remaining software within allocated schedules and budgets based on a cost/schedule analysis of the completed builds cross-compared to the relative risks that emerged from the end-state design analysis. Taken together, these two capstone efforts summarized the overall SoS LCA findings in terms of program performance, not simply software performance, providing management with the "so-what" in a report that was otherwise highly technical.

End-State Design Analysis

The capstone technical analysis was an examination of the overall "end-state" design – the software design as it was expected to exist at the completion of the development program. The analysis sought to determine if the end-state software design would truly meet operational needs of the program. The analysis was an ambitious undertaking; there had never before been an attempt to understand the FCS SoS software design to such a depth. Two problems popped up immediately: 1) how to characterize operational needs in terms of software and 2) how to tie the analysis to those needs.

Rather than recast the program's operational needs in software terms, a more logical approach to the first problem was to analyze the software contributions to the definitive characterization of the program's operational needs—the SoS *key performance parameters* (KPPs). That realization made the analysis approach obvious. The SoS LCA Team applied *assurance cases* to analyze the SoS end-state software design with respect to the KPPs.

Assurance Cases

An assurance case is nothing more than a structured set of arguments, supported by a body of evidence, that justifies belief that a given claim is true. To make one's "case," one argues that certain evidence supports (or does not support) a given claim. One constructs an assurance case by starting with an overarching claim and then iteratively decomposing it into constituent claims, which, at the lowest level, are supported directly by evidence. Figure 4 depicts the concept.

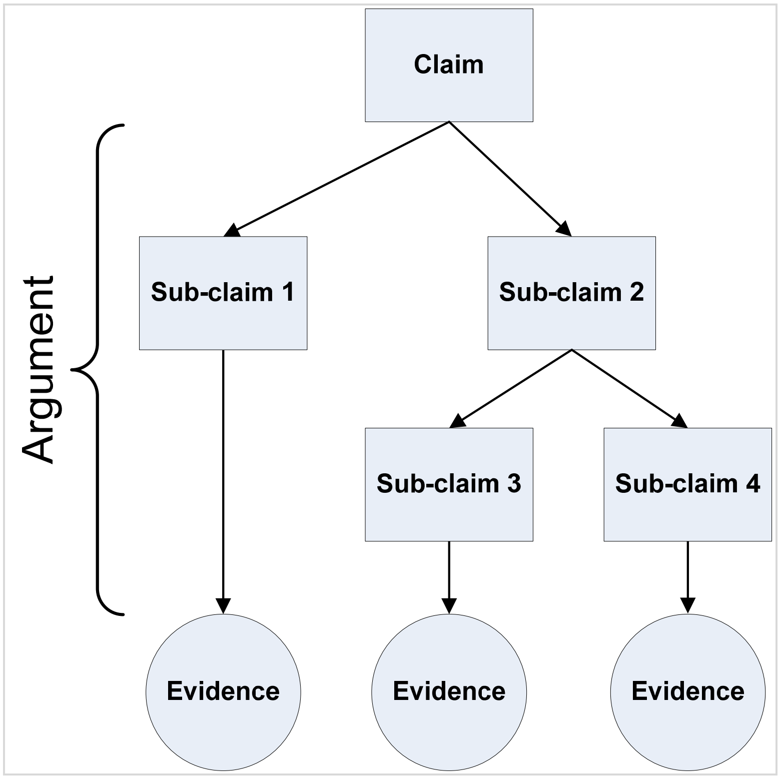


Figure 3. The Basic Structure of an Assurance Case

The logic of an assurance case is straightforward. If all the evidence shown in Figure 4 is sufficient, then belief in sub-claims 1, 3, and 4 is justified. Belief in sub-claims 3 and 4 justifies belief in sub-claim 2, which, combined with belief in sub-claim 1, justifies belief in the top-level overall claim.[2]

Applying Assurance Cases—An Example

For the FCS SoS LCA, the end-state design analysis approach was to use each of the program's KPPs as a main claim and demonstrate how well the SoS software and computing system designs supported them. (In one case, software and computing systems played no role in satisfying a KPP; developing a partial assurance case confirmed that suspicion.) The findings from the focus area analyses served as primary sources of evidence, augmented with additional analyses and artifacts as necessary.

Figure 5 presents an oversimplified example (note that it is not an actual FCS analysis).

[2] A more complete discussion on the use of assurance cases for analyzing SoS software designs can be found in "Assurance Cases for Design Analysis of Complex System of Systems Software" (Blanchette, 2009).

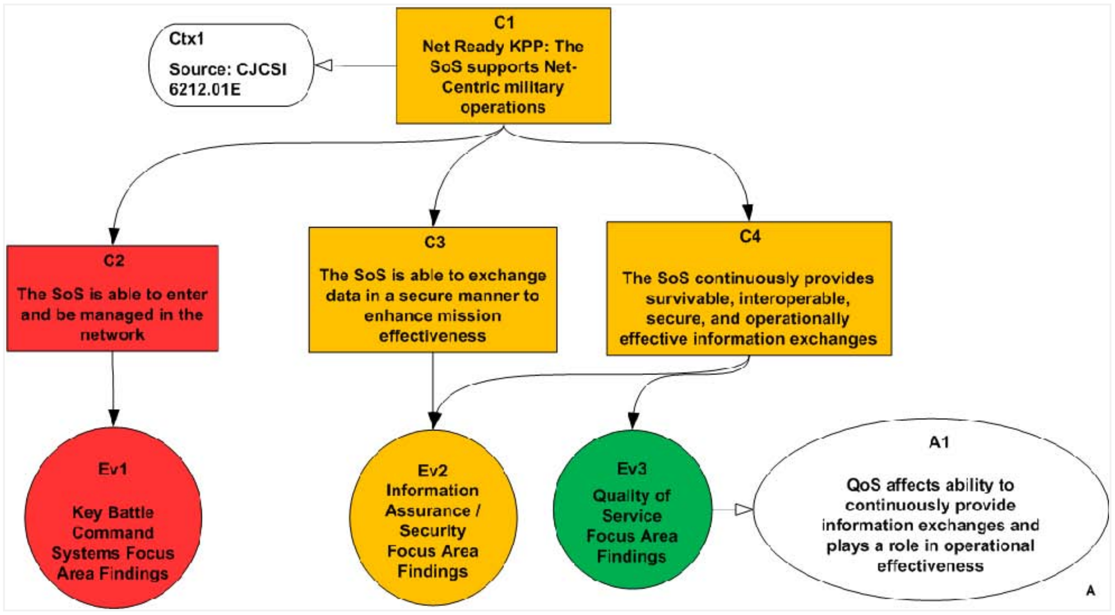

C1
Net Ready KPP: The SoS supports Net-Centric military operations
Ctx1
Source: CJCSI 6212.01E
C2
The SoS is able to enter and be managed in the network
C3
The SoS is able to exchange data in a secure manner to enhance mission effectiveness
C4
The SoS continuously provides survivable, interoperable, secure, and operationally effective information exchanges
Ev1
Key Battle Command Systems Focus Area Findings
Ev2
Information Assurance / Security Focus Area Findings
Ev3
Quality of Service Focus Area Findings
A1
QoS affects ability to continuously provide information exchanges and plays a role in operational effectiveness
A

Figure 4. Oversimplified Example of an Assurance Case Using Focus Area Findings as Evidence

The example shown is based on the Net Ready KPP, a common KPP among DoD systems that exchange information. As shown by the context bubble, Ctx1, the KPP definition is taken from the Chairman of the Joint Chiefs of Staff Instruction (CJCSI) 6212.01E (Chairman of the Joint Chiefs of Staff, 2008) rather than on a specific FCS program requirement. The figure shows the main claim, labeled C1, is supported by three sub-claims, labeled C2, C3, and C4. For purposes of illustration, each sub-claim is shown to be supported directly by findings from the SoS LCA focus areas (Ev1, Ev2, and Ev3).[3] An actual analysis would be far more extensive, with many sub-claims and possibly multiple sources of evidence supporting the claims at the lowest level.

For purposes of communicating findings to management, it is useful to speak in terms of risk (i.e., risk of not meeting a KPP). A common management practice is to associate a color code with different risk levels, such as red to indicate a high level of risk, yellow to indicate a moderate level of risk, and green to indicate a low level of risk. The relative risk levels are then rolled up according to predetermined rules to establish a level of confidence about the supported claims. A sample rule set for this example is shown in Table 2.

[3] In this sample case, assumption A1 helps to justify the use of the QoS focus area findings in support of claim C4 since the connection might not have been obvious.

Table 2. Sample Rules for Rolling Up Risk

Green (low risk)	All lower level claims and supporting evidence are green.
Yellow (moderate risk)	Some lower level claims and supporting evidence are a combination of yellow and red.
Red (high risk)	All, or an overwhelming majority of, lower level claims and supporting evidence are red.

To further illustrate the assurance case concept, the authors have arbitrarily assigned one of each color-coded risk level to the evidence bubbles. Applying the rule set to the example, the findings for key battle command systems indicate a high risk within the focus area (Ev1), so the claim supported by that evidence is also rated as high risk (or, more correctly, low confidence that the claim is true). Applying the rule set to the remaining elements of the example yields the results shown. Since none of the sub-claims in the example is rated any better than moderate risk (indeed, C2 is high risk), the main claim C1 can be no better than moderate risk and, therefore, there is only moderate confidence in satisfying the Net Ready KPP in this example.

A full analysis, of course, would identify the specific issues that led to the risk assignments as well as corresponding recommendations for addressing them. Given such information, a program manager could decide if the moderate risk of not meeting the KPP was acceptable or, if not acceptable, decide on one or more courses of action for reducing the risk/increasing confidence. Although this example is deliberately negative, it should be clear that strong evidence in support of all claims would lead to a high level of confidence that a given KPP was well supported; this situation is, of course, the ideal.

For FCS, the use of assurance cases provided a way to report the technical findings from the SoS LCA to program management in a way that related directly back to program goals without requiring a detailed understanding of software and computing systems. Each KPP had its own assurance case, and the findings from the technical focus areas were used as evidence to support one or more of those assurance cases. This approach tied both issues and strengths in the software to the overall program goals, providing a view of program risk that was vertically integrated within each KPP and horizontally integrated across the technical focus areas.

Producibility Analysis[4]

The second capstone, producibility analysis, concerned the programmatic aspects of the FCS SoS software development effort. Apart from technical feasibility, the producibility analysis sought to demonstrate the feasibility of developing the SoS software and computing systems within cost and schedule targets.

Estimating software producibility presented some special challenges. Unlike hardware, which is typically built to completion before testing, the FCS software was being implemented in a series of increments or builds, which is a fairly common approach for large development programs. In such cases, the software producibility costs for later development increments tend to increase due to breakage in previous increments as well as increased integration and test costs.

[4] Dr. Barry Boehm of the University of Southern California is the principal author of this section.

Breakage in previous increments has several sources. One source is errors discovered in the earlier increments that must be fixed within the budgets and schedules of later increments, which then increases development workload, while the integration and test burden for later increments is already larger due to the software product becoming more complete. Other sources of breakage include revisions of earlier-increment software to support later-increment needs or changes, and adaptation of the software to changes in COTS or external-software interfaces.

The magnitude of these Incremental Development Productivity Decline (IDPD)[5] effects varies due to several factors, such as the degree of coupling to other parts of the software or the rate of requirements or interface changes. Thus, using constant-productivity estimates for future software increment producibility projections would lead to severe underestimation.

Further, FCS was in a unique position in that there were no predecessor projects of equivalent scope and scale from which to predict accurately its software producibility rates. The estimation parameters and knowledge bases of current software estimation tools are generally good for stable, standalone, single-increment development of the kind of software being developed at the lower system levels by the FCS suppliers. However, such tools generally fail to account for the degrees of program and software dynamism, incrementality, coordination complexity, and system of systems integration that were faced by FCS. These phenomena tend to decrease software productivity relative to the cost model estimates, but it is difficult to estimate by how much.

To calibrate software estimates involves measurement of producibility data from early increment software deliveries. For FCS, the best available relevant data was from the System of Systems Common Operating Environment (SOSCOE) software because it underpinned most other mission software packages and thus had several early releases; consequently, there were four increments upon which to base the magnitude and evolution of its IDPD factor. SOSCOE was not fully representative of the mission software, however, leaving uncertainties about its use to predict the IDPD factors of the mission software increments. Similarly, mission software from other programs was not representative of the unique system of systems aspects of FCS. Thus, data from both sources (SOSCOE and other programs) were used to perform a sensitivity analysis of the amount of software that could be produced within the available budget and schedule.

Research Results

Existing software reviews had specific artifacts and criteria to examine, but their scope was necessarily limited. By defining appropriate and relevant criteria at the SoS level, the SoS LCA provided the opportunity to independently assess broader areas that were critical to FCS success. Anchoring the evaluation results with facts and data from early software builds provided a powerful and objective basis for judging the realism of plans for future builds, both from technical and productivity perspectives. Findings were refutable if engineers were able to supply evaluators with additional evidence that had not been

[5] IDPD refers to the phenomenon in which software developer productivity tends to decrease during the later increments of an incremental development project due to a larger code base that must be maintained and integrated in addition to producing new code for the current increment (Boehm, 2009).

considered;[6] indeed, there was one instance in which evaluation findings were revised based on evidence that had not been provided previously. Equally important, the evidence basis of the SoS LCA protected against pessimistic appraisals. In the end, program engineers were largely in agreement with the SoS LCA Team's findings.

The producibility and end-state design capstone analyses (which included roll-ups of findings from the technical focus areas) indicated several places where key changes needed to be made to the FCS SoS software development effort. What is more, regular reports of preliminary findings helped management make adjustments to the program or, in some cases, to the in-progress evaluations. Gaps in design, architecture, and requirements coverage that were unlikely to have been uncovered through other types of reviews were identified and prioritized for correction, providing a path to reach necessary capability levels in time to support a Milestone C decision. Equally important, the SoS LCA provided a mechanism to report technical, cost, and schedule risks, clearly tied to overall program goals, at an appropriate level of detail for senior program management, thereby enabling their understanding and supporting their decision-making processes.

The FCS SoS LCA experience was extremely valuable in providing insights into processes better fitted to the DoD's future systems of systems programs. For example, it has stimulated several improvements to the DoD version of the Incremental Commitment Model (ICM), initially defined by Pew and Mavor (2007) as a process model for DoD human-intensive SoS. The resulting upgrade expressed in its draft DoD version changes the name of the pre-development phase from "Architecting" to "Foundations" (including plans, budgets, schedules, and contract provisions as well as architecture), and changes the name of its phase end-gate from "Lifecycle Architecture" to "Development Commitment Review" (Boehm & Lane, 2008). The SoS LCA also stimulated extension of the ICM process to include guidance for planning, executing, and earned-value managing the development of feasibility evidence as a first-class deliverable. Complementary process improvement initiatives for systems of systems are underway at the SEI.

Some Lessons

FCS conducted the SoS LCA prior to its SoS PDR, which was a good fit for the program; it enabled software to have a relevant voice during an event that otherwise would have been dominated by system and SoS concerns. Had the program continued, the authors believe that a follow-up SoS LCA would have been advantageous prior to the planned CDR as well.[7] In fact, the Brigade Combat Team Modernization (BCTM) program, a follow-on to FCS, is exploring how best to utilize another SoS LCA. In general, the SoS LCA approach described here could be applied at either or both points in the development lifecycle as shown in Figure 6; the key determining factors are the maturity of the SoS architecture and designs, and the availabililty of early implementations to provide an analysis baseline.

[6] Note that such evidence had to have been in existence at the time of the evaluation, not created after the fact to defend against the findings.

[7] In theory, it should be possible to execute a similar type of analyses milestone (called an Initial Operational Capability anchor point in the RUP), again scaled to the SoS level, at the conclusion of the construction phase of development but still prior to initial operational testing, although the need for doing so and the impact of the outcomes are subjects for further research.

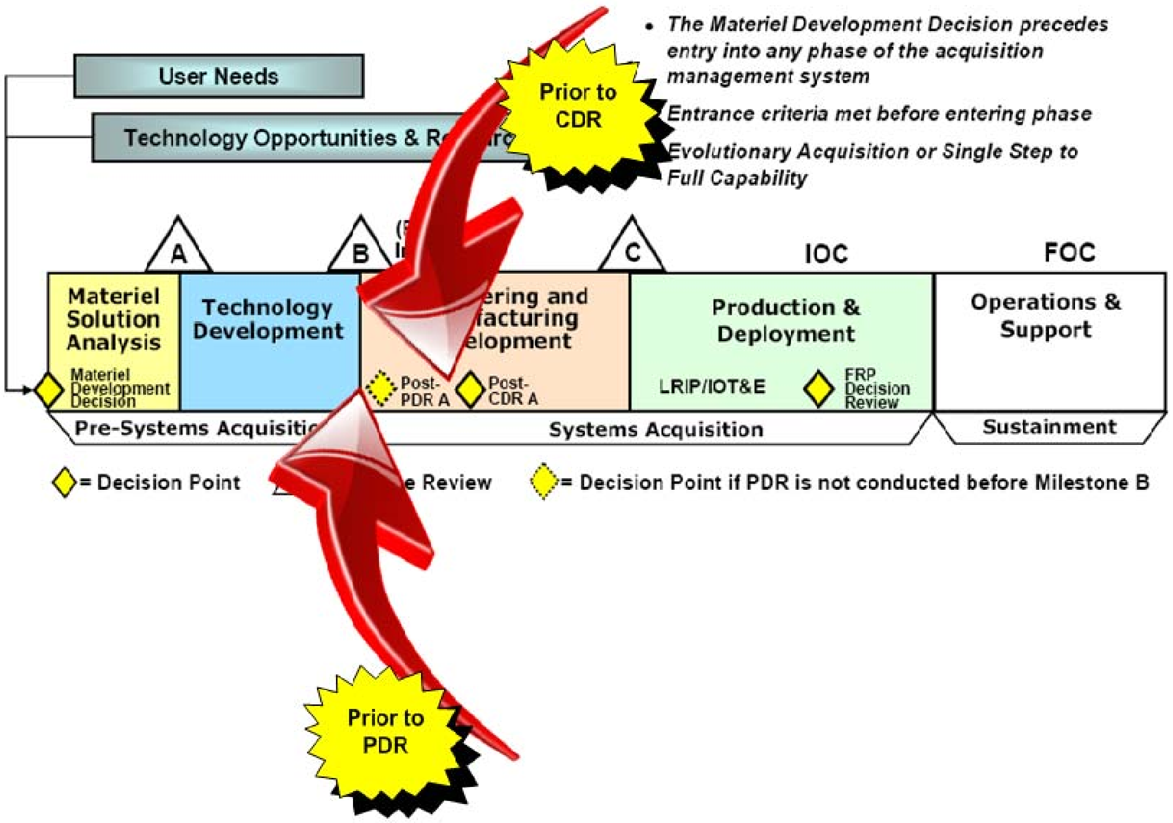

The Materiel Development Decision precedes entry into any phase of the acquisition management system
Entrance criteria met before entering phase
Evolutionary Acquisition or Single Step to Full Capability
User Needs
Technology Opportunities & Resources
Prior to CDR
A
B
C
IOC
FOC
Materiel Solution Analysis
Technology Development
Engineering and Manufacturing Development
Production & Deployment
Operations & Support
Materiel Development Decision
Post-PDR A
Post-CDR A
LRIP/IOT&E
FRP Decision Review
Pre-Systems Acquisition
Systems Acquisition
Sustainment
= Decision Point
= Milestone Review
= Decision Point if PDR is not conducted before Milestone B
Prior to PDR

Figure 5. The SoS LCA Can Be Employed Prior to Traditional Milestone Reviews

Obtaining contractual commitment to have the feasibility evidence produced and the independent evaluation[8] performed is essential. Without such commitment, the SoS LCA Team's requests for data simply did not have sufficient weight when balanced against other program priorities. Once the SoS LCA became a contractual requirement, it was easier for program management at all levels to prioritize requests for information appropriately. In addition, management commitment is indispensable to achieving benefit from an SoS LCA. The true value of the SoS LCA is in being able to make informed program decisions based on facts and data. While the intent is always to conduct an SoS LCA in a way that facilitates decision-making, without true management support the effort (like many other program reviews) is at risk of becoming a "check the box" exercise, where the event is held in order to claim completion on a schedule without regard for the evaluation's effectiveness. Here also, contractual provisions and incentives for developers to plan for and produce feasibility evidence as a first-class deliverable are important elements.

8 Technically, there is no requirement for independent parties to conduct an LCA evaluation. For the FCS program, using independent experts allowed the SoS LCA to be inserted into the development effort with minimal cost and disruption (such as for training staff). Had the SoS LCA been part of the program plans from the outset, independent experts might not have been needed as program staff could have been trained to produce and evaluate the necessary feasibility rationale.

To facilitate inclusion of an SoS LCA in a contract, standards for feasibility evidence should be developed. Such standards would make clear to all parties the kinds of evidence and levels of detail that would be acceptable, enabling the developers to produce it and evaluators to assess it. The FCS program had done this for its lower-level LCA reviews, making the standards a contractual obligation on suppliers. However, the relatively late idea to perform an SoS-level LCA left no time to develop appropriate standards at the SoS level. While this problem was overcome through deeper subject matter analysis, having agreed upon standards would have lessened the burden on the team performing the analyses. Exactly what those standards ought to be is not clear. Surely, the lessons from FCS could be applied to the follow-on BCTM program, but there likely is a generic set of standards that could be extrapolated to fit many different types of SoS.

Gathering artifacts took much longer than anyone had anticipated, during both the pilot and the formal SoS LCA. Part of the difficulty lay in communicating exactly which artifacts were needed and which were available. While the SoS LCA Team believed it communicated its needs clearly, differences in interpretation between the team and program personnel providing the artifacts caused confusion and slowed the process. Another difficulty was that, owing to the sheer size of the program, just finding the custodian who had access to needed information often took a long time.

Keeping management informed of progress and emerging results is essential, but doing so must be managed carefully. As findings became available, status updates to management became more frequent, which in turn led to a decrease in analysis due to the need to spend time to prepare and rehearse briefings and provide read-ahead material in advance of the actual update meetings. Management updates should be scheduled at set points during SoS LCA execution as part of the process.

"Socializing" the process among stakeholders and management in advance is crucial to setting expectations. Many people initially found the SoS LCA concept obscure. Indeed, the term "Lifecycle Architecture" caused a great deal of confusion among managers and project personnel when applied in an SoS context. The word *architecture*, in particular, sets incorrect expectations that the event will be nothing more than an architectural review, which is neither the intent of an LCA review (at any level) nor, as explained earlier, feasible at the SoS level due to the differing development stages among the constituent systems. For this reason, a name other than "LCA" (e.g., Development Commitment Review as used in the Incremental Commitment Model) should be considered while still maintaining the principles embodied in the SoS LCA concept.

Lastly, while the FCS program applied the SoS LCA to software and computing systems only, it should be possible to use the technique on a broadened technical scope encompassing hardware and system issues as well. As a practical matter, it is nearly impossible to ignore such issues even with a limited focus on software.

Conclusion

Extending the concept of an LCA review process to the SoS level was an effective means of evaluating the FCS SoS software and computing system designs. Both the depth and breadth of analysis of this software review far exceeded other software-specific reviews on the program. The broad, multi-build SoS view, in conjunction with the individual build reviews provided an excellent assessment of the state of the FCS software development effort and its potential for achieving program objectives. The technique provided insight into areas of the software development program that had never had an in-depth review.

Overall, the SoS LCA was a success both as a review technique and, more importantly, in providing a solid understanding of the functional baseline for FCS software leading into the SoS PDR. The analyses not only helped discover problem areas, but also recognized software packages that were meeting or exceeding expectations. This latter category was particularly important, as it provided guidelines for recommended paths forward for other software packages. Recommended changes to program processes and evaluations were proposed and being developed for inclusion in the follow-on BCTM program. Rather than mere action items to be tracked to answer specific questions, the SoS LCA provided more details and greater program benefit, including new direction for certain program areas.

The key element of the SoS LCA was the ability to report technical, cost, and schedule risks in a coordinated fashion relative to program goals and *at an appropriate level of detail* for senior program management, thereby facilitating their understanding and supporting their decision-making processes.

The success of the effort suggested possible follow-up activities had FCS continued, including performing one or more “delta” SoS LCA reviews to re-evaluate areas where the highest risks were found and inserting a similar type of effort before a program CDR to gain deeper insights into the development efforts at that stage of the program. While these specific actions were not tried on FCS, it seems reasonable to conclude that they would have been beneficial. This idea is consistent with the concept that shortfalls in feasibility evidence are uncertainties and risks, and should be covered by risk mitigation plans.

The FCS SoS LCA activity also has been a valuable learning experience for preparing and conducting future DoD SoS milestone reviews and acquisition processes.

References

Blanchette, S., Jr. (2009). Assurance cases for design analysis of complex system of systems software. In *Proceedings of the AIAA Infotech@Aerospace Conference and AIAA Unmanned...Unlimited Conference.* Seattle: American Institute of Aeronautics and Astronautics.

Boehm, B. (1996). Anchoring the software process. *IEEE Software*, 13(4), 73-82.

Boehm, B., & Lane, J. A. (2007). Using the Incremental Commitment Model to integrate system acquisition, systems engineering, and software engineering. *CrossTalk*, 20(10), 4-9.

Boehm, B., & Lane, J. A. (2008). *Guide for using the Incremental Commitment Model (ICM) for systems engineering of DoD projects, version 0.5* (USC-CSSE TR-2009-500). Center for Systems and Software Engineering, University of Southern California, Los Angeles, CA. Retrieved from http://csse.usc.edu/csse/TECHRPTS/by_author.html#Boehm

Boehm, B. (2009). Future challenges for software data collection and analysis. In *Proceedings of the Predictor Models In Software Engineering (PROMISE) 2009*, http://promisedata.org/pdf/2009/keynote.pdf

Chairman of the Joint Chiefs of Staff, (2008). *Interoperability and supportability of information technology and national security systems* (CJCSI 6212.01E). Washington, DC: Retrieved from http://www.dtic.mil/cjcs_directives/cdata/unlimit/6212_01.pdf

Kroll, P, & Kruchten, P. (2003). *The Rational Unified Process made easy: A practitioner's guide to the RUP*. Boston: Addison-Wesley.

Pew, R. W., & Mavor, A. S. (Eds.). (2007). *Human-system integration in the system development process*. Washington, DC: National Academies Press.

2003 - 2010 Sponsored Research Topics

Acquisition Management

- Acquiring Combat Capability via Public-Private Partnerships (PPPs)
- BCA: Contractor vs. Organic Growth
- Defense Industry Consolidation
- EU-US Defense Industrial Relationships
- Knowledge Value Added (KVA) + Real Options (RO) Applied to Shipyard Planning Processes
- Managing the Services Supply Chain
- MOSA Contracting Implications
- Portfolio Optimization via KVA + RO
- Private Military Sector
- Software Requirements for OA
- Spiral Development
- Strategy for Defense Acquisition Research
- The Software, Hardware Asset Reuse Enterprise (SHARE) repository

Contract Management

- Commodity Sourcing Strategies
- Contracting Government Procurement Functions
- Contractors in 21st-century Combat Zone
- Joint Contingency Contracting
- Model for Optimizing Contingency Contracting, Planning and Execution
- Navy Contract Writing Guide
- Past Performance in Source Selection
- Strategic Contingency Contracting
- Transforming DoD Contract Closeout
- USAF Energy Savings Performance Contracts
- USAF IT Commodity Council
- USMC Contingency Contracting

Financial Management

- Acquisitions via Leasing: MPS case
- Budget Scoring
- Budgeting for Capabilities-based Planning

- Capital Budgeting for the DoD
- Energy Saving Contracts/DoD Mobile Assets
- Financing DoD Budget via PPPs
- Lessons from Private Sector Capital Budgeting for DoD Acquisition Budgeting Reform
- PPPs and Government Financing
- ROI of Information Warfare Systems
- Special Termination Liability in MDAPs
- Strategic Sourcing
- Transaction Cost Economics (TCE) to Improve Cost Estimates

Human Resources

- Indefinite Reenlistment
- Individual Augmentation
- Learning Management Systems
- Moral Conduct Waivers and First-tem Attrition
- Retention
- The Navy's Selective Reenlistment Bonus (SRB) Management System
- Tuition Assistance

Logistics Management

- Analysis of LAV Depot Maintenance
- Army LOG MOD
- ASDS Product Support Analysis
- Cold-chain Logistics
- Contractors Supporting Military Operations
- Diffusion/Variability on Vendor Performance Evaluation
- Evolutionary Acquisition
- Lean Six Sigma to Reduce Costs and Improve Readiness
- Naval Aviation Maintenance and Process Improvement (2)
- Optimizing CIWS Lifecycle Support (LCS)
- Outsourcing the Pearl Harbor MK-48 Intermediate Maintenance Activity
- Pallet Management System
- PBL (4)
- Privatization-NOSL/NAWCI
- RFID (6)

- Risk Analysis for Performance-based Logistics
- R-TOC AEGIS Microwave Power Tubes
- Sense-and-Respond Logistics Network
- Strategic Sourcing

Program Management

- Building Collaborative Capacity
- Business Process Reengineering (BPR) for LCS Mission Module Acquisition
- Collaborative IT Tools Leveraging Competence
- Contractor vs. Organic Support
- Knowledge, Responsibilities and Decision Rights in MDAPs
- KVA Applied to AEGIS and SSDS
- Managing the Service Supply Chain
- Measuring Uncertainty in Earned Value
- Organizational Modeling and Simulation
- Public-Private Partnership
- Terminating Your Own Program
- Utilizing Collaborative and Three-dimensional Imaging Technology

A complete listing and electronic copies of published research are available on our website: www.acquisitionresearch.org

THIS PAGE INTENTIONALLY LEFT BLANK

A Technique for Evaluating Complex System of Systems Designs

13 May 2010

Stephen Blanchette, Jr.

Software Engineering Institute/Carnegie Mellon University

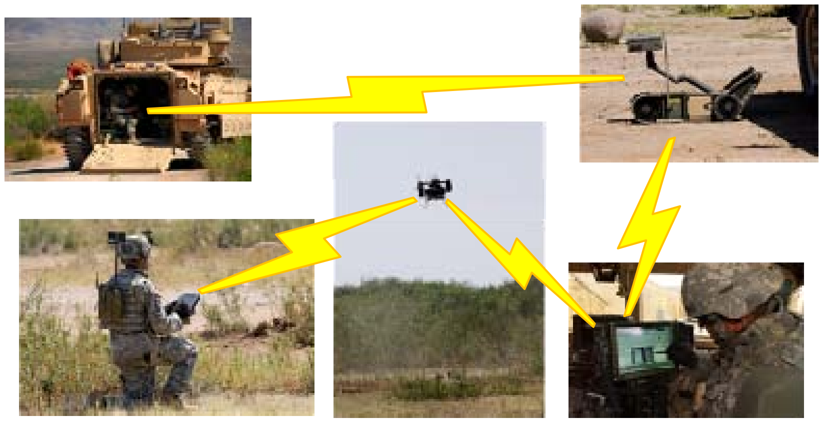
Steven Crosson

US Army – Program Executive Office Integration

Complexity Challenges Our Understanding

DoD Systems are Increasingly Complex…

…Systems of Systems (SoS) even more so



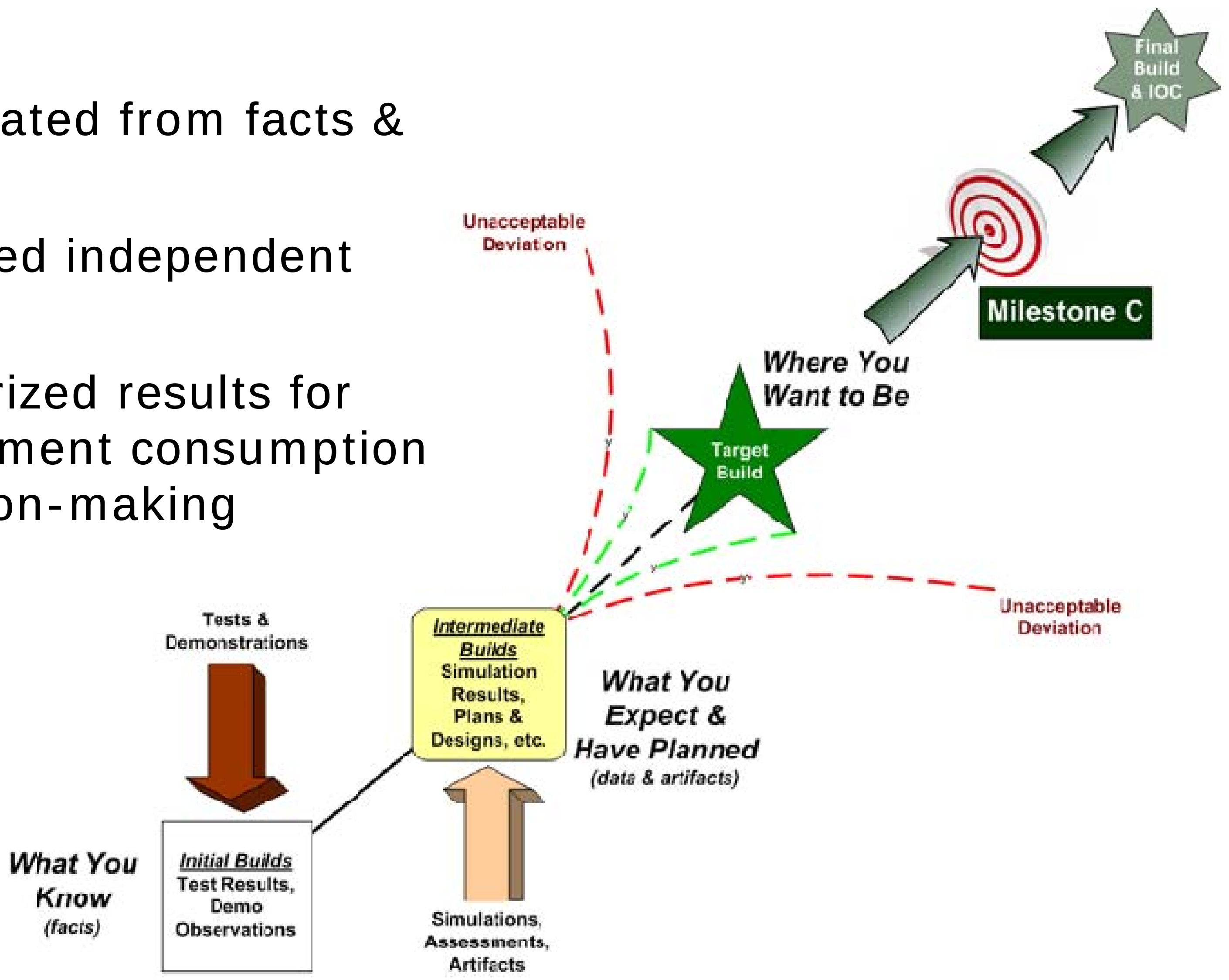
More and more, software drives system/SoS complexity and is the dominating factor in interoperability

Existing Review Types are Inadequate

- PDRs/CDRs tend to focus too narrowly
  - on a mash-up of individual system capabilities rather than true SoS capabilities
  - on functionality at the expense of suitability
  - on work to date rather than on work remaining
  - on PowerPoint artifacts rather than actual data
- Needed: an evidence-based SoS-Level evaluation looking across systems and projecting across builds
- Solution: Lifecycle Architecture (LCA) evaluation
  - LCA demonstrates feasibility of proceeding to construction phase of development
  - Originally a software notion for single systems, had to adapt to SoS
    - Detailed analyses in critical, cross-cutting, technical focus areas
    - Capstones: End State Design & Producibility Analyses

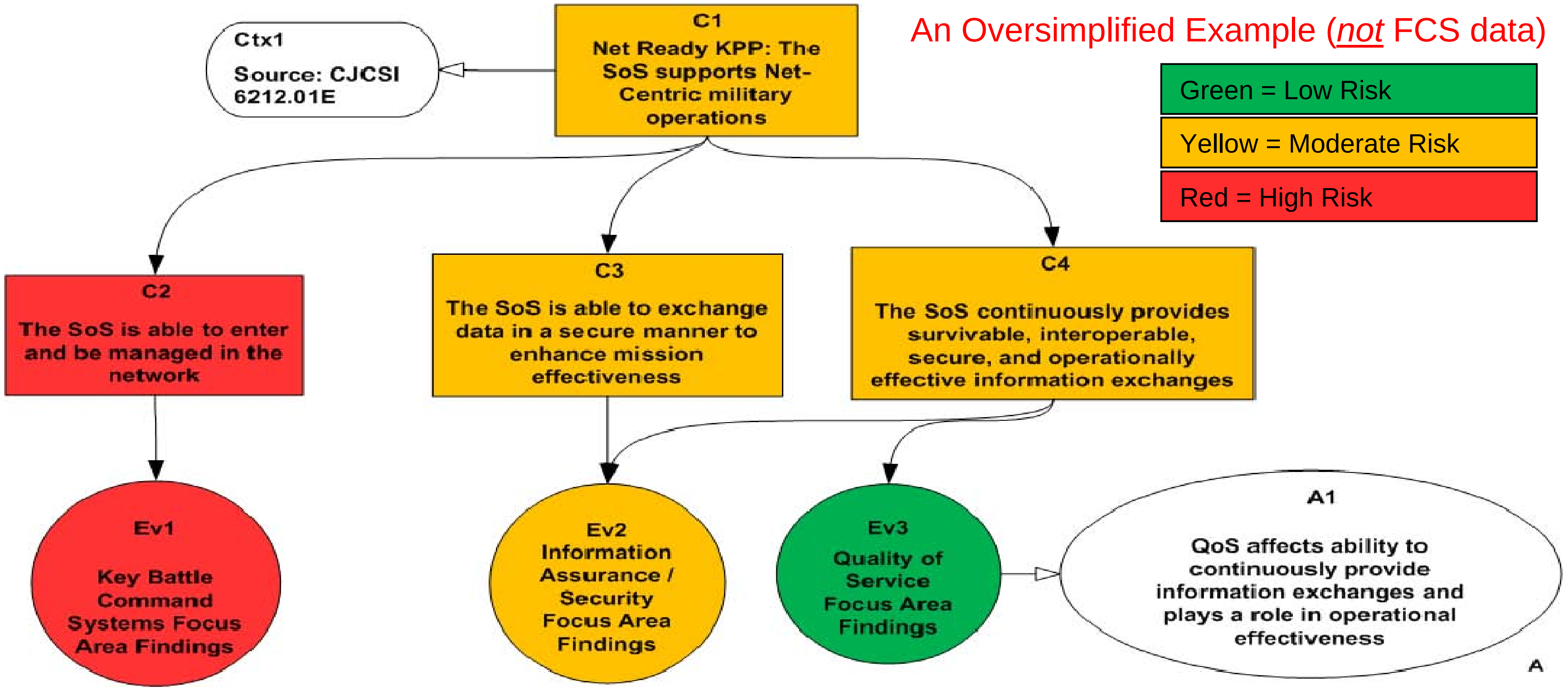
SoS LCA Provided a Better Perspective

- Extrapolated from facts & data
- Leveraged independent experts
- Summarized results for management consumption & decision-making



End-State Design Analysis Made Findings Relevant

- *Assurance Cases* tied technical findings in software to program KPPs
  - Related findings to operational needs
  - Expressed results to aid management decision-making


Ctx1
Source: CJCSI 6212.01E
C1
Net Ready KPP: The SoS supports Net-Centric military operations
An Oversimplified Example (*not* FCS data)
Green = Low Risk
Yellow = Moderate Risk
Red = High Risk
C2
The SoS is able to enter and be managed in the network
C3
The SoS is able to exchange data in a secure manner to enhance mission effectiveness
C4
The SoS continuously provides survivable, interoperable, secure, and operationally effective information exchanges
Ev1
Key Battle Command Systems Focus Area Findings
Ev2
Information Assurance / Security Focus Area Findings
Ev3
Quality of Service Focus Area Findings
A1
QoS affects ability to continuously provide information exchanges and plays a role in operational effectiveness
A

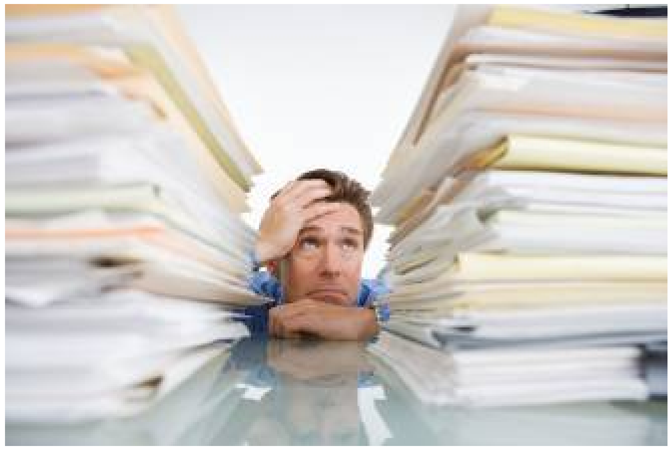
Approved for public release; distribution is unlimited. Case 10-1026. 21 April 2010.

Producibility Analysis Completed the Picture

- Showed feasibility of developing SoS software within cost and schedule targets
  - Factored in Incremental Development Productivity Decline (IDPD)
    - Assuming constant productivity levels would have led to severe underestimation
  - Calibrated estimates based on early builds of SOSCOE and data from other large programs
- Related technical risks to cost & schedule risks

Rework from previous builds…

IDPD Says…


…increases workload & decreases productivity in future builds

SoS LCA is a Useful Tool

- SoS LCA depth/breadth of analysis exceeded other reviews
  - Provided excellent assessment of FCS software development and its potential for achieving program objectives
  - Provided insight into areas of the software development program that had never had an in-depth review
  - Provided management with a previously unseen perspective through use of actual data and fact-based projections rather than confident assertions
- Key was ability to report technical/cost/schedule risks relative to program goals at appropriate level of detail
  - Facilitated management understanding and decision-making
  - Allowed for in-stride program adjustments
- It should be possible to apply the SoS LCA technique to examine hardware/system issues from SoS perspective
  - As a practical matter, these issues are nearly impossible to ignore even with restricted focus on software

The SoS LCA is a means for evaluating and understanding complex Systems of Systems