Software Engineering Institute

Evaluating the Software Design of a Complex System of Systems

Stephen Blanchette, Jr., Software Engineering Institute
Steven Crosson, United States Army
Barry Boehm, PhD, University of Southern California

January 2010

TECHNICAL REPORT
CMU/SEI-2009-TR-023
ESC-TR-2009-023

Acquisition Support Program

This report is based upon work done for Program Manager Future Combat Systems (PM FCS). The program has since been restructured as Program Executive Officer - Integration (PEO I). This report is not necessarily reflective of current PEO I strategy or execution.

http://www.sei.cmu.edu

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

Table of Contents

List of Figures

List of Tables

Acknowledgments

The work described herein was the product of a team effort. The authors are indebted to the dedicated professionals who worked to adapt the concepts and spirit of the Lifecycle Architecture (LCA) anchor point to a large, system of systems (SoS) program. In particular, the authors recognize the following individuals, in alphabetical order, for their significant assistance in developing and executing the process:

- A. Winsor Brown, University of Southern California
- Ed Colbert, University of Southern California
- Robert Ellison, Software Engineering Institute
- Linda Esker, Fraunhofer Center for Experimental Software Engineering, University of Maryland
- Akbar Khan, United States Army
- Jo Ann Lane, University of Southern California
- Derek Lee, Software Engineering Institute
- Ray Madachy, University of Southern California
- Bryce Meyer, Software Engineering Institute
- Anthony Nguyen, The Boeing Company
- Marilyn Phillips, Software Engineering Institute
- Charles Weinstock, Software Engineering Institute
- James Wessel, Software Engineering Institute
- William Wood, Software Engineering Institute
- Carol Woody, Software Engineering Institute

The authors gratefully acknowledge Ms. Cecilia Albert of the Software Engineering Institute (SEI) for her review of this report. Her insightful comments resulted in a much-improved finished product. Mr. Gerald Miller, also of the SEI, provided technical editing assistance that contributed importantly to the quality of this report.

Additionally, the authors thank Major General John R. Bartley, Program Executive Officer, Integration, and Mr. Michael J. Harrigan, Executive Director of Systems Integration for PEO Integration, for authorizing the public release of this report. Sharing successes and lessons learned is crucial to improving the success of future Army and Department of Defense acquisition programs.

Lastly, the authors thank Mr. Edgar Dalrymple from the Software Engineering Directorate of the U. S. Army Aviation and Missile Research Development and Engineering Center and formerly the Future Combat Systems Associate Director for Software and Distributed Systems Integration, whose idea to conduct an SoS LCA was the genesis of this work.

Abstract

Schedule- or event-driven reviews are a crucial element of any major software development project. Such reviews tend to focus on different aspects of development, and different types of reviews provide different benefits. The sum of these reviews, however, is inadequate to address the needs of software development in a complex system of systems (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.

This report discusses the application of the Lifecycle Architecture (LCA) event to what was an enormously complex SoS program: the Army's Future Combat Systems. From the FCS experience, readers will gain insight into the issues of applying the LCA in an SoS context and be ready to apply the lessons learned in their own domains.

1 Introduction

Schedule- or event-driven reviews are a crucial element of any major software development project. 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. Most overall-systems reviews focus mainly on the system-artifact functional definitions, and relatively lightly on evidence that the artifacts described will meet the system's key performance parameter requirements.

These different types of reviews provide different benefits and, taken together, they provide valuable insights into the software development project at various stages of the life cycle. The sum of these reviews, however, is inadequate to address the needs of software development in a complex system of systems (SoS) environment, which is a significant difficulty since systems of systems are becoming almost de rigueur solutions in U.S. defense acquisition programs. In order to deliver orders of magnitude improvement in capabilities for the Warfighter, defense programs increasingly are weaving together existing systems into systems of systems (what is sometimes called an acknowledged SoS) or building a new SoS from scratch (a directed SoS).[1] While systems of systems can deliver unprecedented capability, they also present a number of technical and management challenges.

The notion of an SoS is an abstract concept, often without a physical manifestation that represents the cognitive whole, in which a collection of systems taken together and operating in concert with each other provides greater functionality than the sum of the parts. The dynamic interaction of the constituent pieces yields emergent behavior.[2] Thus, an SoS adds complexity to the typical acquisition program by introducing the need to manage many pieces, each a complex system in its own right, in addition to a somewhat abstract end product.

One of the difficulties with an SoS approach is that while interfaces may be known in advance, how those interfaces may be used to deliver a given capability within the SoS construct under any given set of conditions is not necessarily known. For example, in a networked fires scenario, a sensor may detect and report a target, and a set of algorithms may determine which of several available shooters is best able to engage the target; it is not known in advance which shooter will be selected. This emergence of behavior makes it difficult to have confidence in one's understanding of the contribution of any component-level design to the higher aggregation; small, seemingly non-critical changes can have subtle and unrecognized impacts to other components or to overall SoS performance.

[1] The related taxonomy of SoS was originated in a report by Mark Maier [Maier 1999] and extended to include Acknowledged SoS in a recent *CrossTalk* article [Dahmann 2008]. A summary may be found in a 2009 SEI report [Bergey 2009].

[2] For thorough discussions of emergent behavior, see related reports by John Holland [Holland 1998] and David Fisher [Fisher 2006].

Thus, traditional software review approaches fall well short when used as a means for understanding and evaluating software at an SoS level. What is needed, then, is a method for evaluating software at the SoS level, in addition to the more traditional means of reviewing component-level software designs. What is needed is a true SoS-level evaluation capable of providing an overall assessment of, and insight into, the software development effort in that context—one that can provide a review/assessment of how the developed software capability enables the program's desired operational capability.

The Future Combat Systems program, which was a very large and complex SoS development effort undertaken by the U.S. Army, faced just such a challenge, and hypothesized that something akin to a Lifecycle Architecture (LCA) milestone conducted at the SoS level might be the answer. According to the Rational Unified Process, the LCA marks the conclusion of the elaboration phase of software development, when the requirements baseline is set and the architecture is complete [Kroll 2003, Boehm 1996]. As extended to systems engineering in works by Richard Pew [Pew 2007] and Barry Boehm [Boehm 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. This helps to ensure that program goals now and in the future can be satisfied. The LCA, in particular, strives to ensure that a system, as architected, can be constructed within planned cost and schedule.

Applying the LCA approach to a system of systems introduces some interesting challenges; just as an SoS is not merely a roll-up of its constituent systems, so it is that an SoS-level LCA is more than a roll-up of lower level LCA anchor points. A roll-up approach would fail to capture all of the interesting interactions that underpin the SoS. Further, the nature of SoS programs causes different elements to be at different stages of development at any given time. Therefore, the notion that, at the SoS level, the architecture is complete (and, implicitly, that further development has not yet begun) is unsound.

This report looks at the adaption and application of the Lifecycle Architecture milestone to the software and computing elements of the former Future Combat Systems program. Section 2 provides some background information about the LCA anchor point and the FCS program to set the context. Sections 3 and 4 discuss the LCA anchor point as it was developed and applied to FCS. Section 5 discusses general outcomes of the LCA approach. Section 6 reviews some lessons learned from the experience, and Section 7 offers some conclusions. From the FCS experience, readers will gain insight into the issues of applying the LCA in an SoS context and be ready to apply the lessons learned in their own domains.

2 Background

2.1 The Lifecycle Architecture Milestone

The LCA milestone represents a point in a program where stakeholders come together to assess the work that has been completed through the elaboration phase (the phase in which requirements and architecture models are largely completed) and evaluate the risks of moving forward into the construction and transition phases (the phases in which design, implementation, and testing/validation are performed). The name derives from the event-oriented nature of the review: it is a life-cycle look-ahead, 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. 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 additional work.

The key to the LCA is the 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. The evidence should be objective and consistent within itself to justify the confidence of all stakeholders in moving forward into the latter phases of the development life cycle. Thus, it is important to understand that the LCA is not simply a technical assessment, but 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 moving forward have been identified and sufficiently mitigated to ensure a reasonable chance of success.

The LCA feasibility rationale is relatively straightforward in the context of a traditional system development program. 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 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.

2.2 Future Combat Systems

The FCS program was envisioned by the U.S. Army as a family of manned and unmanned weapon systems, linked by a sophisticated, wireless, battlefield network (see Figure 1).[3] Additionally, the FCS platforms would have been interoperable with currently fielded weapon systems, creating a future fighting force with unprecedented and unparalleled situational awareness and understanding, operational synchronization, and connectivity throughout the joint battlespace [PM FCS 2008]. This requirement to be part of a larger force also created a major challenge for FCS to maintain interoperability with numerous independently evolving external systems.

[3] In June 2009, the Honorable Ashton Carter, undersecretary of defense for acquisition, technology, and logistics, directed the cancelation of the FCS program and its restructure into smaller development projects.

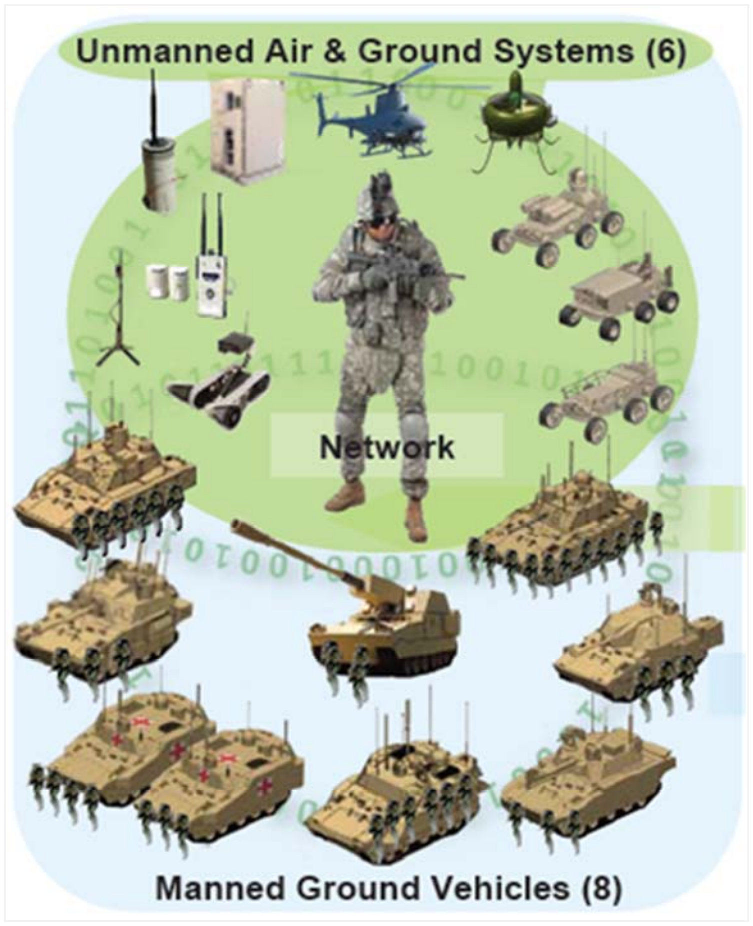

Figure 1: FCS System of Systems [FCS 2008a]

As shown in Figure 2, FCS software is pervasive throughout the SoS, providing a host of battle command applications to the vehicle platforms at the highest level, and, through the FCS System of Systems Common Operating Environment (SOSCOE) software, tying those vehicles to external sensors and to lower network communications services for accessing the Global Information Grid (GIG).

FCS was executed by a Lead Systems Integrator (LSI) team composed of the Boeing Company and Science Applications International Corporation (SAIC). The LSI assembled a best-of-industry team of prime contractors and suppliers, collectively referred to as One Team Partners (OTPs).

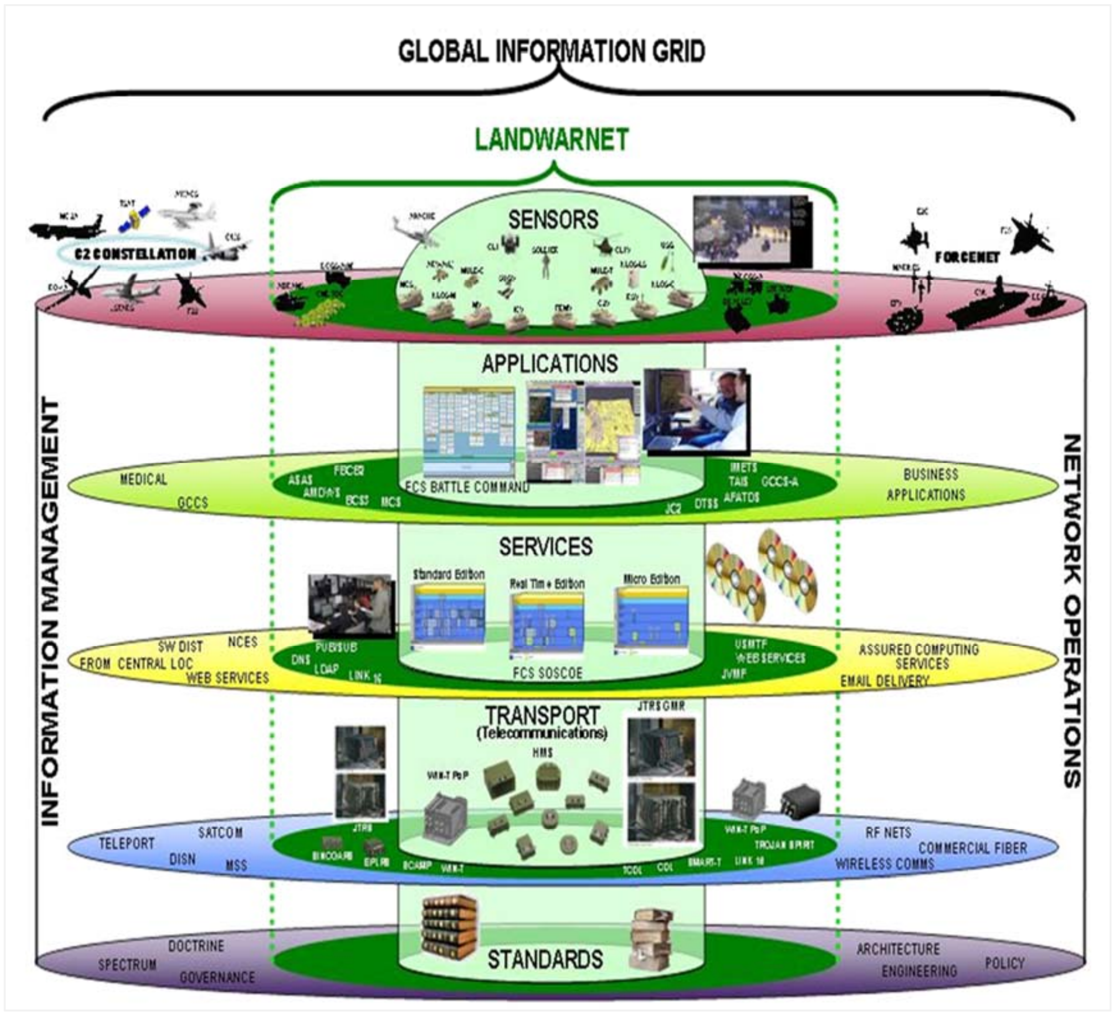

Figure 2: Software is a Key Enabler of the FCS Concept and the Global Information Grid [FCS 2008b]

While the core FCS program focused on producing its main fourteen systems and the network to tie them together, the program also established a technology bridge to the Army's Current Force. Shown in Figure 3, as FCS technologies matured, they were planned for introduction to the Current Force through a series of *spin outs*, allowing the Current Force to become increasingly capable as it evolved toward the Future Force and also ensuring it would be interoperable with new FCS systems as they are fielded.

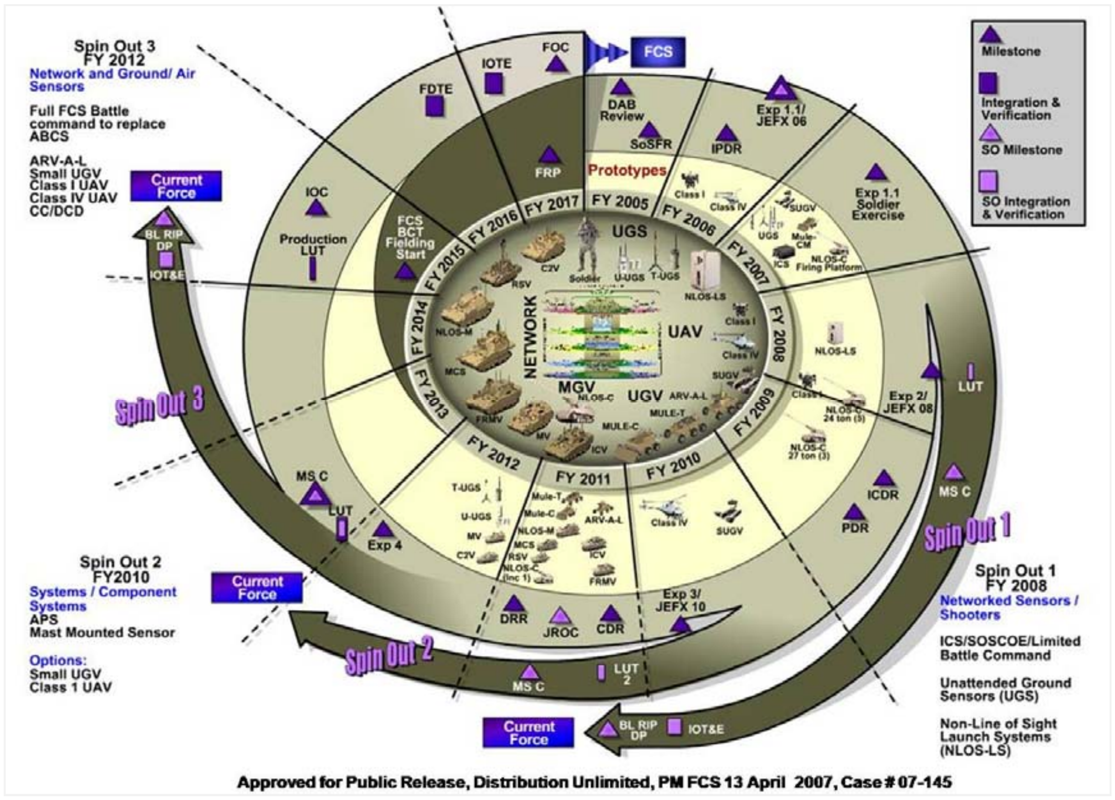

Figure 3: FCS Core Program and Spin Outs

The complexity of the program clearly demonstrated the need for ensuring SoS-level stakeholder commitment to moving forward at key development milestones.

Software for the program was developed in a series of incremental builds, each with its own set of review events having a specific area of concern:

- Evaluation of functionality to be developed against cost and schedule resources
- Evaluation of delivery, integration, and test timelines and criteria
- Evaluation of requirements and interface maturity
- Evaluation of horizontal and vertical integration of capabilities and requirements

For each software build, two levels of review events existed, one level for individual software packages and another for integrated builds.

Although LCAs were conducted regularly at constituent systems levels, the focus of those events was necessarily, and appropriately, build-specific and functionality-specific. The existing reviews, based almost entirely on artifacts, provided important insight into the development plans and results of individual software builds, but tended to focus on evaluating design, architecture, and requirements to develop a solution that *should* meet program needs; comparatively little emphasis was placed on modeling, simulation, and test evidence to show whether the resultant software *would* meet program needs. None of the events provided insight into the entire FCS software effort or how that effort offered a demonstrated contribution to the operational needs. At the SoS level, a broader view was called for; one that considered cross-system functionality across multiple builds to develop confidence that the desired end state would be both reachable and satisfacto-ry.

Although the need for an LCA-like anchor point at the SoS level was apparent, and the LSI's Software Development Plan included a Data Item Description for a Feasibility Rationale docu-ment, it was not clear to what extent this applied contractually across the program. As it had many times before, the FCS program broke new ground, this time by defining its SoS LCA procedures.

3 Defining the SoS LCA Evaluation Process

The first step in the process for defining an SoS-level LCA anchor point was to set up a team of experts from academia, hereinafter referred to as the *SoS LCA Team*, with representatives from the Army providing guidance and representatives from the 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 typical LCA process. They also brought the degree of independence necessary to assure both the LSI and the Army of an unbiased result.

As mentioned earlier, the FCS program had been executing LCAs at the supplier and first-tier integration levels, so there was no need to start from scratch. However, simply rolling-up the results of these lower level reviews would invite missing critical subtleties in the cross-system relationships. For example, a relatively low-risk design change in a battle command system might have unforeseen implications for a logistics system. In addition, the lower level LCA reviews were focused on specific builds, often at different points in time, whereas the SoS-level LCA was obliged to project across builds. Further, 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. Thus, it was necessary to construct an evaluation process that was anchored by the best-available facts and data as the basis for projecting forward.

3.1 Conceptualizing the SoS LCA

From the outset, the SoS LCA definition effort was steered by few key guidelines that helped drive the ultimate solution. These guidelines were:

- 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?
- Base answers on *evidence* rather than opinion.
- Discover issues 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[4] decision.

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 these determinations was objective evidence: engineering work products were evaluated for completeness and adequacy, as well as for consistency among and between 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 project the ability to meet program needs.

4 Milestone C (MS C) is a capstone review in the DoD system acquisition life cycle. Success marks the completion of the System Development and Demonstration (SDD) phase of a program and approves entry into the Production and Deployment (P&D) phase [DAU 2005].

Rather than finding fault and assigning blame, the goal of the 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.

As shown in Figure 4, the FCS SoS LCA 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. The SoS LCA results were a key feeder into the program PDR event.

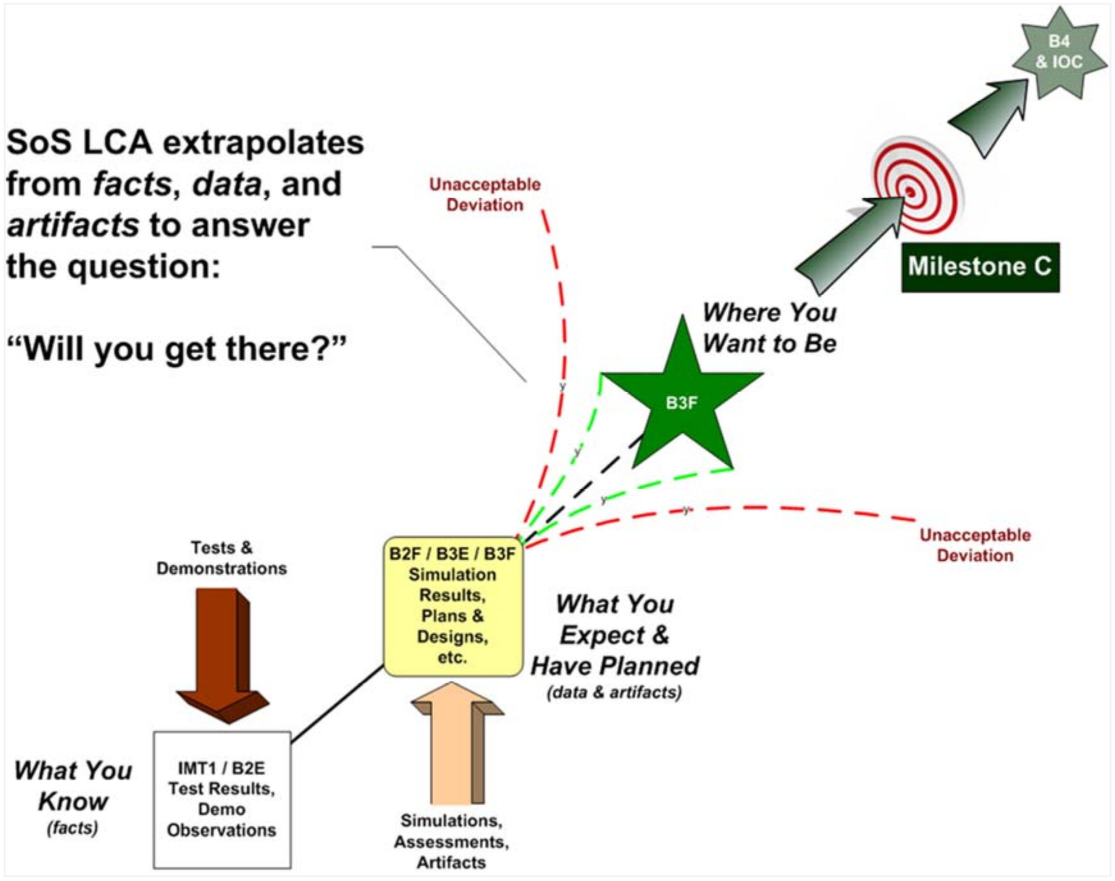
SoS LCA extrapolates from *facts*, *data*, and *artifacts* to answer the question:
"Will you get there?"
B4 & IOC
Milestone C
Unacceptable Deviation
Where You Want to Be
B3F
Unacceptable Deviation
Tests & Demonstrations
B2F / B3E / B3F Simulation Results, Plans & Designs, etc.
What You Expect & Have Planned
(data & artifacts)
What You Know
(facts)
IMT1 / B2E Test Results, Demo Observations
Simulations, Assessments, Artifacts

Figure 4: SoS LCA Determines the Likelihood of Achieving an Acceptable Outcome

The basis of the extrapolation was *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, etc. that formed the basis of the work yet to be done.

3.2 Focus Areas

The SoS LCA Team's first challenge was to grapple with scope. On a very large SoS program such as FCS, the scope of the LCA becomes extremely important. 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 become invalid; as development work continues, prior problems or unknowns become resolved while new issues may be discovered.

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. There are practical limitations on how much can be reasonably accomplished at the SoS level, and 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. A detailed understanding of the designs would consume vast resources in terms of time, money, and people. In addition, coordinating an effort of that magnitude would become a project within the project. Such an LCA would be too large, too complex, and too resource-intensive to be managed successfully. Instead, the team had to re-envision what a modest-sized team with limited resources could reasonably accomplish. This realization naturally led to 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.

The original ten high-payoff areas postulated by the SoS LCA Team based on the Army sponsors' priorities and the team members' knowledge of the program were:

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

A cursory review of the results from lower level LCAs revealed that these focus areas were commonly cited as problems. With the focus areas thusly defined, the SoS LCA Team turned its attention to developing the SoS LCA process.

3.3 The Initial Process Model

Having established the scope and focus areas for the SoS LCA, the SoS LCA Team next turned its attention to the evaluation process itself. The initial SoS LCA process model is shown in Figure 5. As envisioned, several analysis teams consisting of experts from the SoS LCA Team and representatives from the government, LSI, and OTPs were to be formed. Each team would tackle a single focus area, examining the relevant facts, data, evidence, and plans and developing conclusions. 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 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.

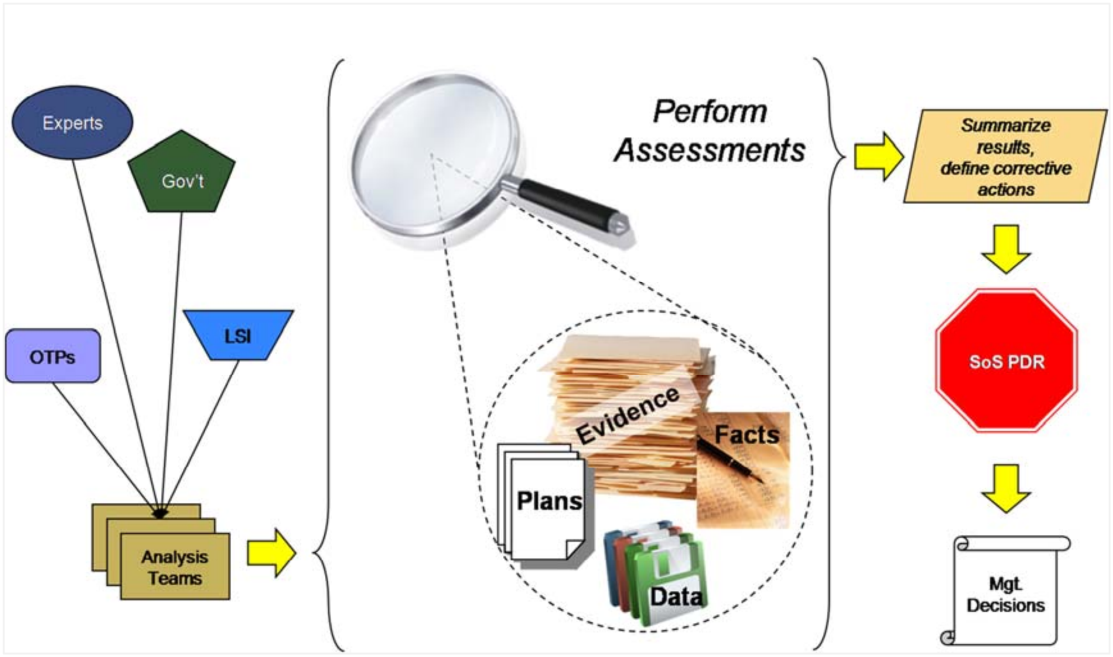

Figure 5: The Initial SoS LCA Process Model

Members of the SoS LCA Team divided the focus areas among themselves based on their respective backgrounds and expertise.

3.4 Piloting the Initial Process

Once the initial process model was in place, the next challenge was to figure out the details of its execution. The team determined that a small-scale pilot would be the best way to refine the process. The intent of the pilot was to prove out the process steps in a single area before attempting to scale the process up to the full SoS.

In some respects, planning the pilot was as complicated as planning the process itself. The principle difficulties were determining the scope and timing of the pilot. In terms of scope, it was crucial to limit the extent of the pilot so that it was executable in a short timeframe. At the same time, the scope needed to be broad enough to be useful as a test run for the larger effort. No less daunting was the challenge of 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 LCA pilot. The need to minimize disruption to ongoing program activities served to complicate matters further.

Developing relationships with key program personnel turned out to offer avenues of approach for these challenges. The lead of the C4ISR[5] Team volunteered his area of responsibility to participate in the LCA pilot because he saw value in having some independent review of the work being performed by his team. His area of work, however, was enormous. Thus, scoping became the first order of business. Adjustment of the pilot scope to be achievable yet worthwhile was accomplished through negotiations with the team lead. The agreed-upon emphasis centered on a subset of the 10 focus areas, coupled with the C4ISR lead's principle areas of concern. The timing of the pilot was propitious; the C4ISR Team had a review cycle already planned for the March 2007 timeframe, aligning nicely with the desired LCA pilot time frame.

The pilot itself consisted of a few parts. The first step was to socialize the idea among the C4ISR Team members so that expectations on both sides were clear. The next step was to gather data. The SoS LCA Team spent about three weeks gathering evidence from existing C4ISR plans and technical documents. The SoS LCA Team was very explicit that no new documents were to be created for the purposes of the pilot.

One early discovery during the pilot effort was that SoS LCA Team members were not in full agreement about the goals or 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 of the pilot process was that despite efforts to clearly set expectations, there was considerable misunderstanding between the SoS LCA Team, the C4ISR Team, and its various performer organizations. Documentation and evidence furnished by the C4ISR Team led to more questions than answers, and required several follow-up teleconferences to request additional documents. The documents provided by the performer organizations 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 of performing a complete assessment.

Actual evidence of feasibility was lacking. That is not to say that the C4ISR Team had not done its job, as the contracts were unclear about what the scope of "the job" of providing feasibility evidence was. Rather, there was little documentation to support an independent conclusion that the C4ISR plans and approach were feasible. That is the crux of the LCA; it relies upon evidence rather than verbal accounts and assurances. Part of the problem was that the pilot was focused too broadly, looking at several areas of C4ISR that had not been planned for development during Build 1, which is when the pilot was executed.

Despite these difficulties, the pilot was hugely helpful in clarifying the SoS LCA Team's thinking about conducting the SoS LCA. It demonstrated that program personnel would be too busy to participate significantly in gathering and examining feasibility evidence. It also demonstrated the need to set expectations among developers and management early and often to ensure their buy-in to, and cooperation with, the process. The pilot also highlighted the need to review emerging findings with stakeholders to ensure accuracy. The C4ISR Team also benefitted from the pilot by be-

5 C4ISR stands for command, control, communications, computers, intelligence, surveillance, and reconnaissance.

ing able to raise issues that had been of concern to them for management attention with the added weight of independent analysis to support them.

3.5 Refining the Process

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 to represent the essential FCS capabilities, the "non-disposable" elements of the program. These elements were ones that had to work in order to ensure that the program produced 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

Focus Area	Description
End-State Design	Determination of whether the end-state design will meet operational needs. This includes identification of risks/issues and recommendations to mitigate/address them
Software Producibility	Analysis of FCS software against cost and schedule models with acknowledgement that the scope and size of the FCS software effort is larger and more complex than most other projects

Of particular note are the last two focus areas: end-state design and software producibility. End-state design was a capstone technical focus area that summed up the findings of the other technical focus areas in a way that made the findings relevant for management decision-making. It was added as a focus area midway through the analysis process when the SoS LCA Team realized that the other technical focus areas were yielding results at too low a level to support strategic decision-making. Refer to Section 4.3 in this report for additional details about end-state design. Software producibility looked specifically at the feasibility of developing the remaining software within allocated schedules and budgets based on cost/schedule analysis of the early builds and also on the relative risks that emerged from the end-state design analysis. These two areas together summarized the overall SoS LCA findings in terms of program performance, providing the "so-what" in a report that was otherwise highly technical. The other focus areas formed the detailed backup in the final report to management.

Figure 6 shows the changes to the process model, 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). The realities of personnel availability necessitated these slight adjustments.

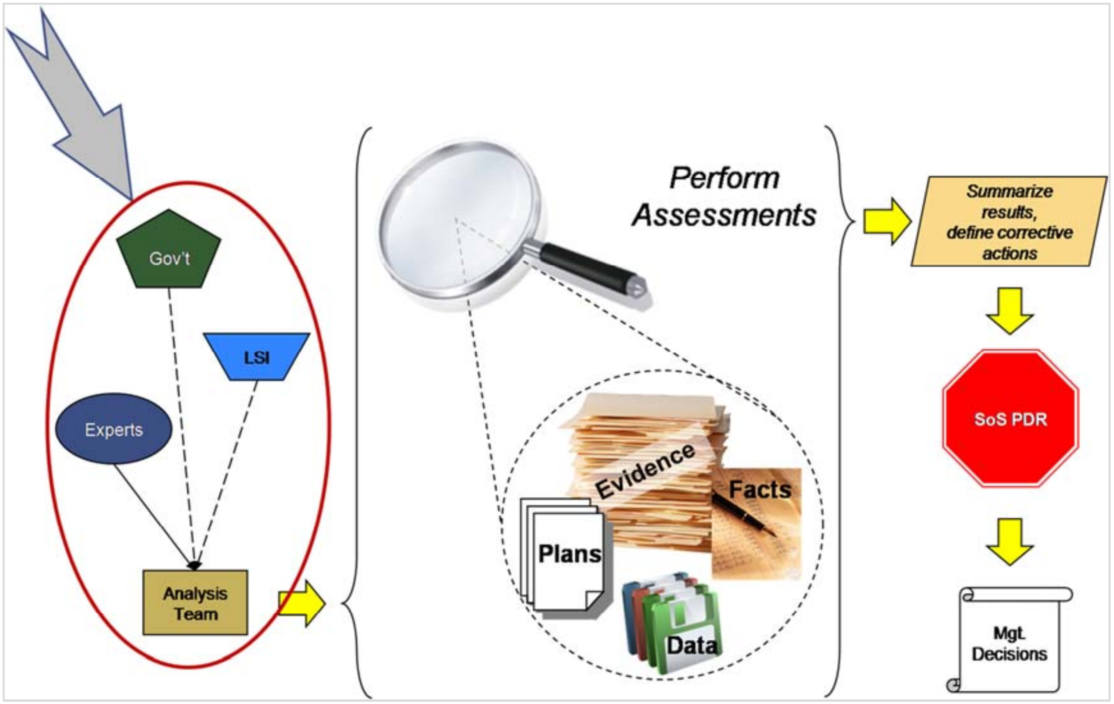

Figure 6: SoS LCA Process Model Showing Refinements

As shown in Figure 6, instead of several analysis teams, there was only one analysis team composed entirely of technical experts from the Software Engineering Institute, the University of Southern California, and the Fraunhofer Center for Experimental Software Engineering at the University of Maryland. Rather than providing full-fledged analysis team members, the Army was able only to provide guidance while the LSI was only able to facilitate access to documentation. OTPs were unable to participate in the analysis process at all, although they were called upon to provide data and, occasionally, clarification of the data. Focus areas were assigned to one or two team members who were then responsible for identifying the types of artifacts needed for evaluation, reviewing them, and rendering recommendations based on review findings. The entire analysis team would be responsible for reviewing results as they emerged.

3.6 Scheduling the SoS LCA

The final step in defining the SoS LCA process was determining where to insert it into the overall program schedule. The SoS LCA was not part of the original program plan, so it was not reflected in the program's master plan or master schedule. For a typical single-system LCA, the execution point is nominally at the gateway between the elaboration and construction phases of the program. For a complex SoS program, such a distinction is less meaningful, as the various elements of the program move between those phases at different points in time. Thus, the FCS program had to find a schedule point that was the logical equivalent of the gateway to the construction phase, but at the SoS level.

The answer was to conduct the SoS LCA prior to the program PDR, the point at which the overall SoS design would have reached a sufficiently mature, but not yet fully detailed, state. This location seemed to be a "natural" fit for the SoS LCA in the FCS schedule. Tying the SoS LCA to the program PDR also provided a very convenient opportunity to present the results to program management in context with other SoS technical details. Further, the program PDR needed a component that addressed software specifically, as preparations had focused almost exclusively on SoS hardware and external interaction concerns. As a practical matter, completion of the SoS LCA and delivery of the final report were added to the conditions for successful closeout of the FCS program PDR.

4 Performing the SoS LCA Evaluations

Having established a fixed endpoint for completion, the SoS LCA Team worked backwards to schedule the technical evaluations, allowing time at the end to draft a final report to management. To build program management confidence that the SoS LCA would be executable, the team decided on a two-phased execution approach: a Preliminary Analysis phase and a Detailed Analysis phase.

4.1 Preliminary Analysis

The Preliminary Analysis phase examined the focus areas where data was believed to be most available. This phase was designed to give the SoS LCA Team insight into the quantity and quality of data available for analysis, to give program leadership sufficient opportunity to provide feedback on the execution of the SoS LCA, and to allow the OTPs and LSI early insights into the potential need for corrective action. It also yielded insights to the SoS LCA Team into the ongoing evaluation. For instance, the SoS LCA Team realized that some minor focus areas could be combined into meta-focus areas (such as was the case for Fusion and Data Distribution), some focus areas, such as Display Management and Task Automation/TINs, needed no further evaluation during in the Detailed Analysis phase, and one focus area, namely Software Performance, was missing from the plan.

4.2 Detailed Analysis

The Detailed Analysis phase expanded upon the preliminary analyses to explore each focus area in depth. The goal was to identify specific strengths and weaknesses in each focus area along with specific, actionable recommendations for corrective action where needed. The SoS LCA Team also met regularly to discuss emerging findings since the results from one focus area often had bearing on other focus areas.

In addition to findings within each focus area, at the conclusion of this phase, the SoS LCA Team formed summary recommendations. These recommendations were deemed *major*, meaning that there would be significant benefit to the program when implemented, or *common*, meaning that the recommendations were applicable to more than one focus area.

4.3 Technical Capstone: End-State Design Analysis

The capstone technical focus area was an examination of the overall “end-state” design, that is, the software design as it was expected to exist at the completion of the development program. The focus area sought to determine if the end-state software design would truly meet operational needs based upon the available data, documentation, and plans. The analysis was an ambitious undertaking; there had never before been an attempt to understand the 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 analyze a still-evolving SoS software design.

Rather than figuring out how to put 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).[6] With that realization, the analysis approach then became obvious. The SoS LCA Team elected to use *assurance cases* to analyze the SoS end-state software design with respect to the KPPs.

An assurance case, simply, is a structured set of arguments supported by a body of evidence that justifies belief that a given claim is so. To make one's "case," one argues that certain evidence supports a given claim. As a practical matter, 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 by direct evidence. Figure 7 depicts the concept.

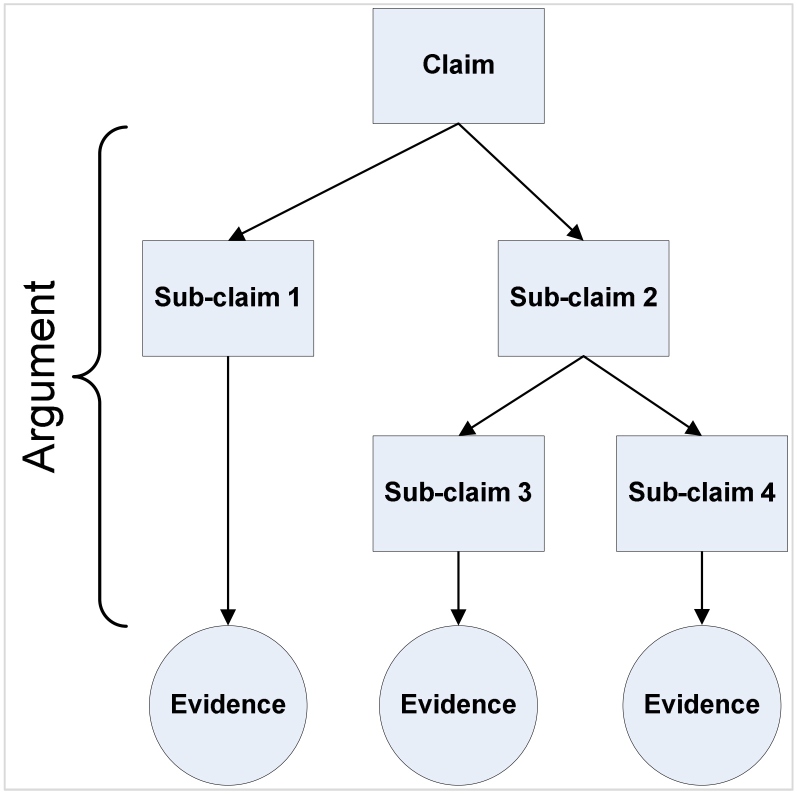

Figure 7: The Basic Structure of an Assurance Case

The logic is straightforward. If the evidence is sufficient, then belief in sub-claims 1, 3, and 4 is justified. Belief in sub-claims 3, 4 justifies belief in sub-claim 2, which, combined with belief in sub-claim 1, justifies belief in the top-level overall claim.[7]

For FCS, then, 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. Figure 8 presents an oversimplified example (note that it is not an actual FCS analysis).

6 Within the U.S. Department of Defense (DoD), KPPs are an expression of the system characteristics considered to be essential for delivery of an effective military capability [CJCSI 2007]. All DoD projects have some number of KPPs to satisfy in order to be considered acceptable from an operational perspective.

7 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].

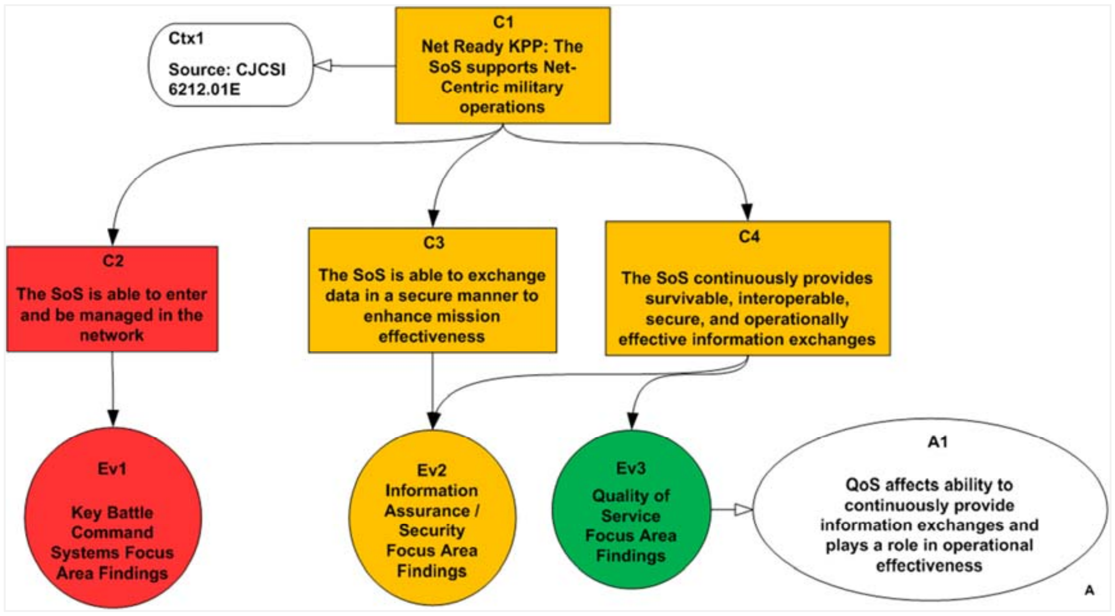
Ctx1
Source: CJCSI 6212.01E
C1
Net Ready KPP: The SoS supports Net-Centric military operations
C2
The SoS is able to enter and be managed in the network
C3
The SoS is able to exchange data in a secure manner to enhance mission effectiveness
C4
The SoS continuously provides survivable, interoperable, 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 8: Oversimplified Example of an Assurance Case Using Focus Area Findings as Evidence

The example shown in Figure 8 is based on the Net Ready KPP, a common KPP among DoD systems that must exchange information. The example is drawn from the KPP definition found in the Chairman of the Joint Chiefs of Staff Instruction (CJCSI) 6212.01E [CJCSI 2008] (as shown by the context bubble Ctx1) rather than on a specific FCS program requirement. The figure shows the main claim, C1, is supported by three sub-claims C2, C3, and C4. For purposes of illustration, each sub-claim is shown to be directly supported by findings from the SoS LCA focus areas (Ev1, Ev2, and Ev3).[8]

To further illustrate the concept, the authors have arbitrarily assigned color-coded risk levels to the evidence bubbles as follows: red indicates a high level of risk within a focus area, yellow indicates a moderate level of risk, and green indicates 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.

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.

In this example, since the findings for key battle command systems indicate a high risk within the focus area (Ev1), the claim supported by that evidence is also rated as high risk (or, more correctly, low confidence that the claim is true). Since none of the sub-claims in this example is rated any better than moderate risk (indeed, C2 is high risk), the main claim C1 can be no better than moderate risk. Hence, one would say that there is only moderate confidence in satisfying the Net

8 In this case, assumption A1 helps to justify the use of the QoS focus area findings in support of claim C4.

Ready KPP. A full analysis, of course, would identify specific issues and recommendations at each level. Armed with this information, a program manager could decide if the moderate risk of not meeting the Net Ready 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 claims would lead one to a high level of confidence that a given KPP was well supported.

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 knowledge 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.

4.4 Producibility Analysis

The last of the focus areas was concerned with 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, software is often implemented in a series of increments or builds. The software producibility costs for later development increments tend to increase due to previous increment breakage as well as increased integration and test costs.

This breakage has several sources. One is that errors discovered in earlier increments must be fixed within the budgets and schedules of later increments, which increases development workload, while the integration and test burden for later increments is already larger due to the software product becoming more complete. Others 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)[9] 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 by the FCS One Team Partners. However, such tools generally fail to account for the degrees of program and software dynamism, incrementality, coordination complexity, and system of systems integra-

9 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].

tion 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 SOSCOE software because it underpinned most other software packages and thus had several early releases. This had the benefit of an early start, since it was needed to support the mission software; thus, there were four increments upon which to base the magnitude and evolution of its IDPD factor. It was not fully representative of mission software, leaving uncertainties about its use to predict the IDPD factors of the mission software increments. But mission software from other programs was not representative of the unique system of systems aspects of FCS. Thus, data from both sources were used to perform a sensitivity analysis of the amount of software that could be produced within the available budget and schedule. Continuing efforts have been underway to assess the IDPD factors of various classes of mission software as its increments are produced.

5 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.

Rooting results in facts and data from early software builds provided a powerful basis for judging the realism of plans for future builds, both from technical and productivity perspectives. Refutation of the findings was possible if engineers were able to supply evaluators with additional evidence that had not been considered[10] and, indeed, there was one instance in which evaluation findings were revised based on evidence that had not previously been provided. Equally important, the evidence basis of the overall analysis protected against pessimistic appraisals. In the end, program engineers were largely in agreement with the SoS LCA Team's findings.

Analysis of the producibility and end-state focus areas (including the roll-ups of other focus areas) indicated several areas where key changes should 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 LCA experience has been 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 from its initial definition as a process model for such future DoD human-intensive systems of systems in [Pew 2007]. The resulting upgrade expressed in its draft DoD version in [Boehm 2008] changes the pre-development phase name 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. It also stimulated extension of the 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.

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

6 Some Lessons Learned

Ideally, the SoS LCA should have been part of program planning from the outset. However, since the concept was not fully built into contract provisions and occurred after software development activities began, such was not an option in the FCS case. The effort to develop the process, achieve buy-in from stakeholders, and execute the analyses would have been greatly reduced had the necessary contract provisions and resources, such as requirements to conduct an SoS LCA and explicit definition of feasibility evidence expected as an output of the engineering work, been built in from the outset. Such resources are needed both for the developers to produce feasibility evidence and for the independent evaluators to assess it.

Obtaining contractual commitment to have the feasibility evidence produced and the independent evaluation 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.

Starting the SoS LCA process definition too early is a definite danger. On FCS, the SoS LCA Team started work before a definite date for the SoS LCA had been determined. Due to this floating schedule, and given all the other program priorities, it was difficult for the SoS LCA Team to maintain focus and commitment. Even tying the SoS LCA to the program PDR was not sufficient to build team focus; initially, the planned date of the PDR was too far in the future for team members to expend any serious energy on defining or executing the SoS LCA. A period of about a year and a half seems to have been ideal, with about six months for defining the process and planning the event and 12 months to execute the necessary process steps. The amount of assessment effort would be considerably less to the extent that the developers' contracts and budgets supported development of the feasibility evidence.

It is vitally important to maintain close coordination within the evaluation team, especially when that team is geographically dispersed. The SoS LCA Team also faced an additional coordination challenge in that members came from different organizations and thus had different tools for video teleconferencing and collaboration. In such an environment, face-to-face meetings were the only way to achieve effective brainstorming sessions, accomplish detailed planning, and consolidate evaluation findings for management-level consumption. Weekly teleconferences are helpful in keeping the effort moving, but the most significant progress occurred when team members were able to meet in person.

A lesson from the pilot (and confirmed during the formal SoS LCA) was that it takes much longer to gather artifacts than anyone had anticipated. This was true despite the fact that all the artifacts were already in existence. 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.

Providing feedback to management is essential, but it has to be managed carefully. The initial findings from the preliminary analysis phase generated a great deal of management interest. It also spurred more frequent updates, which in turn led to an increase in “viewgraph engineering” and a corresponding decrease in analysis due to the time needed to prepare and rehearse briefings as well as to 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.

The SoS LCA Team started out with a rating scheme that was too fine. The initial scheme had five levels, which presented a problem when trying to report early results from the initial phase of the evaluation to program management. Management strongly preferred so-called “stop light charts” showing green for low risk areas, yellow for moderate risk areas, and red for high risk areas. Trying to force the results of a five-level rating system into a three-level reporting system resulted in one of two awkward solutions: shades of red/yellow/green or inconsistency of ratings within each level. In either case, the message became lost in the medium, as more time had to be spent on explaining the rating system than the actual ratings.

Management commitment is essential 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” activity where the event is held simply for the sake of being able to claim completion on a schedule. Here again, contractual provisions and incentives for developers to plan for and produce feasibility evidence as a first-class deliverable are important to obtaining management commitment.

“Socializing” the process among stakeholders and management in advance is crucial to setting expectations. Many people initially found the SoS LCA concept obscure. Even when conducting the pilot with an organization that had been through lower level LCAs, there was not a good understanding of the types of evidence sought.

The term “Lifecycle Architecture” seems to cause a great deal of confusion among managers and project personnel when applied in an SoS context. Particularly, use of the word *architecture* sets incorrect expectations that the event will be nothing more than an architectural review, which is neither the intent of an LCA at any level nor, as explained earlier, feasible at the SoS level due to the differing stages of development among the constituent systems. Thus, anyone planning to adopt the SoS LCA technique described in this report might wish to consider an alternate name while still maintaining the principles embodied in the LCA concept.

7 Conclusion

The SoS LCA 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 event far exceeded other software-specific review events on the program. The broad, multi-build SoS view, in conjunction with the individual system or individual software build reviews provided an excellent assessment of the state of the FCS software development effort. New areas of analysis 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 new piece to the FCS software review puzzle and, more importantly, in providing a solid functional baseline for FCS software leading into an 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 restructured 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 *at an appropriate level of detail* for senior program management, thereby enabling their understanding and supporting their decision-making processes.

The success of the effort suggests possible follow-up activities, including performing one or more "delta" SoS LCA events 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 have not been tried on FCS, it seems reasonable to conclude that they would be 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 has also been a valuable learning experience for preparing and conducting future DoD SoS milestone reviews and acquisition processes.

Feedback

Through its Acquisition Support Program, the SEI is working to help improve the acquisition of software-intensive systems across the U.S. government. As part of its mission, the SEI is very interested in learning how other organizations are tackling the challenges of developing, managing, or acquiring systems of systems. In addition, the SEI is pleased to discuss the information in this report in more detail.

Please send questions or comments about this report to Stephen Blanchette, Jr. (sblanche@sei.cmu.edu).

Appendix Acronyms and Abbreviations

The alphabetical list below contains all acronyms, abbreviations, and their meanings as used in this report.

ABCS	Army Battle Command System
APS	Active Protection System
ARV-A-L	Armed Robotic Vehicle - Assault - Light
B2E	Build 2 Early
B2F	Build 2 Final
B3E	Build 3 Early
B3F	Build 3 Final
B4	Build 4
BCT	Brigade Combat Team
BLRIP DP	B-Kit Low Rate Initial Production Deployment Period
C2V	Command and Control Vehicle
C4ISR	Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance
CC	Centralized Controller
CDR	Critical Design Review
CJCSI	Chairman of the Joint Chiefs of Staff Instruction
CMU	Carnegie Mellon University
COTS	commercial off the shelf

CSSE	Center for Systems and Software Engineering
DAB	Defense Acquisition Board
DCD	Dismounted Control Device
DoD	Department of Defense
DRR	Design Readiness Review
ESC	Electronic Systems Command
FCS	Future Combat Systems
FDTE	Force Development Test and Experimentation
FOC	Full Operational Capability
FRMV	FCS Recovery and Maintenance Vehicle
FRP	Full Rate Production
FY	Fiscal Year
GIG	Global Information Grid
IA	Information Assurance
ICDR	Initial Critical Design Review
ICM	Incremental Commitment Model
ICS	Integrated Computing System
ICV	Infantry Carrier Vehicle
IDPD	Incremental Development Productivity Decline
IEEE	Institute of Electrical and Electronic Engineers

IMT1	Integration Maturity Test 1
IOC	Initial Operational Capability
IOT&E	Initial Operational Test and Evaluation
IOTE	Initial Operational Test and Evaluation
IPDR	Initial Preliminary Design Review
IPT	Integrated Product Team
JEFX	Joint Expeditionary Force Experiment
JROC	Joint Requirements Oversight Council
KPP	Key Performance Parameter
LCA	Lifecycle Architecture
LSI	Lead Systems Integrator
LUT	Limited User Test
MCS	Mounted Combat System
MS C	Milestone C
MULE-C	Multi-function Utility/Logistics and Equipment Vehicle – Countermine
MULE-CM	Multi-function Utility/Logistics and Equipment Vehicle – Countermine
MULE-T	Multi-function Utility/Logistics and Equipment Vehicle – Transport
MV	Medical Vehicle
NLOS-C	Non-Line of Sight Cannon

NLOS-LS	Non-Line of Sight Launch System
NLOS-M	Non-Line of Sight Mortar
NMS	Network Management System
OTP	One Team Partner
P&D	Production and Deployment
PDR	Preliminary Design Review
PM	Program Manager
RSV	Reconnaissance and Surveillance Vehicle
RUP	Rational Unified Process
QoS	Quality of Service
SAIC	Science Applications International Corporation
SDD	System Development and Demonstration
SEI	Software Engineering Institute
SO	Spin Out
SoS	System Of Systems
SOSCOE	System of Systems Common Operating Environment
SoSFR	System of Systems Functional Review
SUGV	Small Unmanned Ground Vehicle
TINs	Task Integration Networks
TR	Technical Report

T-UGS	Tactical Unattended Ground Sensor
U.S.	United States
UAV	Unmanned Air Vehicle
UGS	Unattended Ground Sensor
UGV	Unmanned Ground Vehicle
U-UGS	Urban Unattended Ground Sensor
WMI	Warfighter Machine Interface

References

URLs are valid as of the publication date of this document.

[Bergey 2009]
Bergey, John; Blanchette, Jr., Stephen; Clements, Paul; Gagliardi, Mike; Klein, John; Wojcik, Rob; & Wood, Bill. *U.S. Army Workshop on Exploring Enterprise, System of Systems, System, and Software Architectures* (CMU/SEI-2009-TR-008). Software Engineering Institute, Carnegie Mellon University, 2009. http://www.sei.cmu.edu/library/abstracts/reports/09tr008.cfm

[Blanchette 2009]
Blanchette, Jr., Stephen. “Assurance Cases for Design Analysis of Complex System of Systems Software,” *Proceedings of the AIAA Infotech@Aerospace Conference and AIAA Unmanned...Unlimited Conference*, Seattle, WA, April 2009.

[Boehm 1996]
Boehm, Barry. “Anchoring the Software Process.” *IEEE Software 13*, 4 (July 1996): 73-82. http://www.computer.org/portal/web/csdl/doi/10.1109/52.526834

[Boehm 2007]
Boehm, Barry & Lane, Jo Ann. “Using the Incremental Commitment Model to Integrate System Acquisition, Systems Engineering, and Software Engineering.” *CrossTalk Journal 20*, 10 (October 2007): 4-9. http://www.stsc.hill.af.mil/crosstalk/2007/10/0710BoehmLane.html

[Boehm 2008]
Boehm, Barry & Lane, Jo Ann. “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, 2008. http://csse.usc.edu/csse/TECHRPTS/by_author.html#Boehm

[Boehm 2009]
Boehm, Barry. “Future Challenges for Software Data Collection and Analysis,” Keynote, Predictor Models In Software Engineering (PROMISE) 2009, Vancouver, British Columbia, Canada, May 2009. http://promisedata.org/pdf/2009/keynote.pdf

[CJCSI 2007]
Chairman of the Joint Chiefs of Staff Instruction. “Joint Capabilities Integration and Development System,” CJCSI 3170.01F, May 2007. http://www.dtic.mil/cjcs_directives/cdata/unlimit/3170_01.pdf

[CJCSI 2008]
Chairman of the Joint Chiefs of Staff Instruction, “Interoperability and Supportability of Information Technology and National Security Systems,” CJCSI 6212.01E, December 2008. http://www.dtic.mil/cjcs_directives/cdata/unlimit/6212_01.pdf

[Dahmann 2008]
Dahmann, Judith; Rebovich Jr., George; & Lane, Jo Ann. "Systems Engineering for Capabilities." *CrossTalk 21*, 11 (November 2008): 4-9.
www.stsc.hill.af.mil/crossTalk/2008/11/0811DahmannRebovichLane.html

[DAU 2005]
DAU (Defense Acquisition University). "DAU Glossary of Defense Acquisition Acronyms and Terms, 12th Edition." https://dap.dau.mil/aphome/das/pages/dauonlineglossary.aspx

[FCS 2008a]
Future Combat Systems. *FCS East Coast Regional Conference*. August 2008.

[FCS 2008b]
Future Combat Systems. *Smart Book*. September 2008.

[Fisher 2006]
Fisher, David A. *An Emergent Perspective on Interoperation in Systems of Systems* (CMU/SEI-2006-TR-003). Software Engineering Institute, Carnegie Mellon University, 2006.
http://www.sei.cmu.edu/library/abstracts/reports/06tr003.cfm

[Holland 1998]
Holland, John. *Emergence*: *From Chaos To Order*. Basic Books, 1999 (ISBN: 9780738201429).
www.perseusbooksgroup.com/basic/book_detail.jsp?isbn=0738201421

[Kroll 2003]
Kroll, Per & Kruchten, Philippe. *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*. Boston: Addison-Wesley, 2003 (ISBN-10: 0321166094).
www.informit.com/store/product.aspx?isbn=0321166094

[Maier 1999]
Maier, Mark. "Architecting Principles for Systems of Systems." *Systems Engineering Journal 1*, 4 (February 1999): 267-284. http://www.infoed.com/Open/PAPERS/systems.htm

[Pew 2007]
Pew, Richard & Mavor, Anne (eds.). *Human-System Integration in the System Development Process*. National Academies Press, 2007. http://books.nap.edu/catalog.php?record_id=11893

[PM FCS 2008]
Program Manager, Future Combat Systems (Brigade Combat Team). System Overview. December 2008. www.fcs.army.mil/news/pdf/FCS_White_Paper_DEC_08_Final.pdf

REPORT DOCUMENTATION PAGE		*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE January 2010	3. REPORT TYPE AND DATES COVERED Final	
4. TITLE AND SUBTITLE Evaluating the Software Design of a Complex System of Systems		5. FUNDING NUMBERS FA8721-05-C-0003	
6. AUTHOR(S) Stephen Blanchette, Jr., Steven Crosson, & Barry Boehm			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213		8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-2009-TR-023	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116		10. SPONSORING/MONITORING AGENCY REPORT NUMBER ESC-TR-2009-023	
11. SUPPLEMENTARY NOTES			
12A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS		12B DISTRIBUTION CODE	
13. ABSTRACT (MAXIMUM 200 WORDS)			

Schedule- or event-driven reviews are a crucial element of any major software development project. Such reviews tend to focus on different aspects of development, and different types of reviews provide different benefits. The sum of these reviews, however, is inadequate to address the needs of software development in a complex system of systems (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.

This report discusses the application of the Lifecycle Architecture (LCA) event to what was an enormously complex SoS program: the Army's Future Combat Systems. From the FCS experience, readers will gain insight into the issues of applying the LCA in an SoS context and be ready to apply the lessons learned in their own domains.

14. SUBJECT TERMS SoS, system of systems, life cycle architecture, LCA		15. NUMBER OF PAGES 51	
16. PRICE CODE			
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. Z39-18 298-102