Adjusting Software Life-Cycle Anchorpoints

Lessons Learned in a System of Systems Context

Steven Crosson (PM FCS) and Barry Boehm (USC)

Abstract

Evaluating the state of a software-centric program based on paper-based reviews is always challenging. Predicting future challenges from those reviews is even more so. The Army's Future Combat Systems program has implemented Life Cycle Objective and Life Cycle Architecture events to provide continuous software reviews and risk analysis for individual software products. However, through lessons learned from those events, a new process was implemented for a System of Systems level review.

The review model was expanded to include reviews of individual system test data aimed at evaluating key program risks and providing data-based course corrections. Additionally, in-depth software productivity data was reviewed and analyzed to provide a projection of future capability development and recommendations for software build plan adjustments. This article will identify key changes made to the review process and why these changes, based on lessons learned, provided a more complete assessment of the FCS software effort.

1. Introduction

Any software development effort can benefit from schedule or event-driven reviews. Whether to determine software quality, software functionality, software performance, or any other measure, reviews can help find issues early, or lead developers in new and better directions. The more complex the effort, the more value these software reviews, or checkpoints, can provide. Equally, having several varying reviews with different foci in terms of goals, can increase that value-added.

The FCS software development effort is incredibly complex, and to help make the development tasks more manageable, software components are developed in a series of semialigned “builds.” Each build is planned out to develop certain program functionality and capabilities. While each build has its own set of capabilities to deliver, and are developed at the root level, independently of other builds, the end result of the FCS software development effort (and the combination of all software builds) must be an integrated System of Systems (SoS) and so all builds must be intergrable both vertically and horizontally. To aid in assessing the progress of the program’s software development, scores of metrics are kept and regularly analyzed to determine progress, issue discovery and issue resolution, schedule impacts, etc. These metrics are reviewed at varying levels of detail at review events and checkpoints as defined by the FCS Software Development Plan [FCS SDP, 2007]. While the review events below are build specific, this paper will explore the concept of a SoS-level review event in the pre-PDR (and theoretically, pre-CDR) context: a SoS Life Cycle Architecture event.

Current Review Events

The most widely utilized review events on the FCS program include the Life Cycle Objective (LCO) and Life Cycle Architecture (LCA) events for individual software packages [Boehm, 1996], and the Build Definition Checkpoint (BDC), Build Planning Checkpoint (BPC), Build Readiness Checkpoint (BRC) and Build Assessment Checkpoint (BAC) at the integrated product level (several software packages integrated to form a single “build” of software).

Below is a short summary of the purpose of each of these events. They will be discussed in the order they would occur in the FCS Software Development Lifecycle [FCS SDP, 2007].

- BDC – The main focus of this event is to present, discuss, and agree upon the SoS level build plan (capabilities to be developed in the build the BDC is focused on – Build X, Build Y, etc). Also, the SoS schedule is presented at the BDC to provide context of how this software build fits into the larger FCS software development picture. Key artifacts are presented, such as build requirements, mission scenarios, architecture approaches, and reusable-software evaluations.

- LCO – This review is a de facto kick off of the development of the specific software package that is the subject of the LCO. This event is for the software developers to demonstrate that they understand the capabilities and functionality that must be provided in this software build. Also, they must demonstrate that they understand and have identified at least one software architecture that can be developed to ensure that those required capabilities can be provided. The understanding of these points will be displayed through development of software requirements specifications, top-level architecture views, build plans, schedules and top-level evidence of feasibility.
- BPC – This planning checkpoint consists of a roll-up of all the artifacts developed and discussed in the individual software development team LCO events. Thus, the BPC has a SoS or SoS-like context. The BPC includes reviews of the design, architecture, and risk artifacts not only for the subject build, but within a context of all builds. Horizontal integration across all software development efforts is also emphasized at the BPC.
- LCA – This event focuses on design and prototyping activities to demonstrate feasibility of the specific architecture to be used and of its ability to satisfy the requirements, especially key performance parameters (KPPs). Each software development effort must participate in an LCA event. This review event is considered the equivalent of a high-content, low-ceremony PDR in terms of assessing evidence of product maturity and ability to reach completion within the resources available. Build scope, objectives and test plan artifacts are all reviewed and discussed at the LCA. (It should be noted, that

based on this description of the LCA, the SoS LCA is a bit of a misnomer, this will be explained more later).

- BRC – This is widely considered the most important of the checkpoint reviews on the FCS Software program. This checkpoint is geared to serve as a commitment, based on evidence that the needed software capabilities for this build can be developed and tested to the defined requirements within existing budget, schedule, and personnel constraints using the documented design and architecture. This event should integrate all the artifacts from the individual software developer LCA events.
- BAC – The assessment checkpoint is focused on analyzing the integration and test result artifacts, as well as establishing draft lessons learned. This also serves as the review event to examine real versus estimated cost and schedule data.

Evaluation of Current Review Events

It should be noted that while these review events cover a wide range of the software development life cycle, and provide status on key components of any software development effort, that large portions of the reviews are indeed artifact- and evidence-based. There is definite value in artifact-centric, evidence-based reviews as they provide the opportunity to evaluate the software development planning, design and architecture perspectives and help ensure the proper software solution is being developed, hopefully cutting down on errors of interpretation of intent, and resulting in a product that *should* deliver the proper capability.

There is an inherent issue with the checkpoint events with regard to consistency of artifacts. This is especially a problem in the checkpoint events that involve “roll-ups” of LCO and LCA artifacts. While the FCS Software Development Plan does lay out the artifacts to be presented at each LCO and LCA event, there is room for each software development team to create their respective artifacts in slightly different ways. Also, due to the nature of contract adjustment timelines, if there are any adjustments to "official program baselines" in the requirements or other areas, there can be issues that each software development team is conducting their LCO or LCA event with slightly different requirements baselines. This can lead to an inconsistency amongst LCO and LCA artifacts and can make interpreting them at a checkpoint event level more challenging. Thus, there is a need for an over-arching review that encompasses more than just the plans as laid out in the existing program software reviews, a review that can determine "where are we?" as a software program, and help determine if we can get where we need to be to have a successful product.

These existing reviews have less emphasis on review of test and evaluation data. This is due in large part to the timing of the events, which emphasize obtaining assurance the proper product is being built, a much more cost-effective way to diagnose software problems than to find them during test or to develop representative models and simulations to evaluate such KPPs as scalability to large numbers of engaged entities. However, a lot of value can be gained through the review of test and evaluation results. Of chief concern is which requirements and KPPs were tested successfully and which were not, as this helps establish the current software functional and performance baseline for the FCS program. Knowing what capabilities can be operationally executed within mission timelines and other KPPs is often viewed as the ultimate bottom line by the user. As such, having more review of what capability has been evaluated with respect to readiness for use based on representative operational scenarios should hold a place of importance in program software reviews as well.

2. Concept of a SoS LCA

Despite the long list of the benefits of these existing component-level and build-level reviews, there was a desire to understand exactly what ultimate SoS operational capability would be achievable heading into a PDR timeframe. As such, a new event was added to the program schedule, a SoS LCA.

The SoS LCA is a very unique event, in that it is using data, preferably test, modeling, and simulation data, (though in the areas of determining software producibility patterns, cost and schedule data were included) to provide an analysis of what capability the FCS software program is able to produce. From that analysis, the goal of the SoS LCA is to project what capabilities can be developed by FCS, based on operationally-representative architecture-based benchmarks, models and simulations and on historical producibility rates, prior to a Milestone C decision point [FCS SoS LCA Plan, 2008].

This approach differs from the majority of the existing program review checkpoints in that the focus is almost exclusively on existing data, and not on plans or designs (though the plans and designs were examined in conjunction with the data to determine estimated versus actual capability development).

Selection of Areas to Examine

On a program the size of FCS there are is no shortage of crucial software efforts. However, due to the size of the SoS LCA review team and the amount of time to conduct the review, there was a limit to how many various areas/issues could be examined simultaneously. As such, the team tried to select a set of key items and concepts that would be crucial to a large number of the FCS Program Integrated Product Teams (IPTs) in an effort to get the most “bang for the buck.” The selected areas - termed Focus Areas by the SoS LCA team - that were investigated will be discussed individually later. It should be noted that these Focus Areas were not necessarily “risk areas” but just areas crucial to the success of the program.

The important output of the review of these key program software issues was not just to provide a status, but a projection on how the overall program software development effort would be impacted by the status of the Focus Area. This projection, if less than the current program plan, was to be accompanied by recommendations for adjustments, either to the development effort, concept of operations, level of capability, etc., to ensure the most useful capability set was produced based on the current Focus Area status and available resources going forward. These recommendations are the most critical output of the SoS LCA effort, and are where the true value of the review is reflected.

Those recommendations are yet another output unique to this SoS LCA review and will be used to help guide our software development effort, both at a program and individual software package level, to ensure the most capability possible is delivered in the software builds leading up to the Milestone C decision. The other review checkpoints capture action items that must be addressed, but those action items are focused on local vs. global SoS effects. By developing recommended SoS-level action plans, and having their execution tied to completion of SoS-level rebaselining activities, the results of the SoS LCA review will produce measurable and configuration controllable local outputs that come together to produce the desired global SoS effects.

<u>SoS LCA Focus Areas</u>

The following areas were reviewed by the SoS LCA review team:

- *Software Producibility*: The FCS software is larger and more complex than previous projects to which current software cost and schedule estimation models have been calibrated. It is also being developed and integrated in multiple increments by numerous organizations, which phenomena are also incompletely modeled by current parametric estimation models.

- *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 Productivity/Performance*: Analysis of the processes for developing and testing software products, their effectiveness in testing requirements, and of the performance efficiency of the software.
- *Information Assurance and Security*: Review of the status of efforts to develop and test FCS IA components, and a determination of the attainment of necessary functional capabilities
- *Distributed Information Management*: Effectiveness of data communications between platforms and systems over the FCS network, analysis of data reliability, latency, etc
- *Distributed Fusion Management*: Analyze effectiveness of utilizing several sources to gather and distribute fusion data, includes network usage, data reliability, latency, accuracy, etc.
- *Network Management*: Review of current status of development of the Network Management system, as well as analysis of the current state of general network management issues.
- *Modeling and Simulation*: Analysis of the process used to develop modeling and simulation test environments and supplemental functionality. Determine issues facing this process and determine methods to mitigate those issues.

<u>Lower Level Areas of Analysis</u>

The above list captures the high level Focus Areas that the SoS LCA team was to investigate, but does not offer an all-encompassing list of the areas addressed through this review activity. In the analysis of these top-level areas, several key, lower level, software building blocks will have to be investigated to provide a complete analysis.

- *Interfaces* – By the nature of investigating areas such as security, information distribution, external interoperability, and software performance, interfaces have to be part of the equation. The definition of interfaces, testing of interface requirements, and adequate understanding of the capabilities the interface will provide by parties developing on both ends is critical to any software effort, but is especially key when examining the fidelity of over-arching software needs. If an interface is improperly defined or doesn't function to its requirements, then none of those over-arching needs will be met. Net-centric systems such as FCS have much more complex interface specifications than traditional definitions of interface message content. They must also define such protocols as joining the network, presenting credentials, security handshakes, publishing capabilities, subscribing to other capabilities, and exception handling.
- *Hardware Availability and Performance* – Any analysis of software on the FCS program must come with an implicit review hardware availability and performance, as that availability is one factor that dictates when software capability can be developed. Without the hardware to run on, the software could not be tested in similar to fielded conditions and so any assessment of recommended paths forward must take into account hardware timelines and allocations.

- *Lessons Learned* – Through reviewing already completed test and/or simulation events, the review team has the ability to utilize not just the data from the event, and its implications, but also the lessons learned. Whether these be lessons learned in regards to setup, initialization, etc they can provide valuable input to the analysis as well as any recommendations the review team develops.
- *Integration Efforts* – The use of test data also provides the opportunity to review how well various pieces of FCS software (as well as hardware) are integrating. This ties back to the interfaces and hardware availability items above, but also goes deeper. This includes having the right personnel available for the integration, the proper amount of time allocated for the integration activities, the proper integration facilities, scenario generators, instrumentation, and data analysis capabilities, and a satisfactory integration plan to be followed to ensure success.

Additional Benefit: Review of Overall Program Needs

By virtue of having to narrow down a list of focus areas the review team would have the resources to analyze, the SoS LCA provided a de facto method of determining the key FCS program software needs. Just the performing of that action alone was the first value that the SoS LCA provided that was unique with respect to the other software checkpoints. Those other reviews have pre-determined sets of criteria and artifacts to be produced and reviewed, but having a blank page for the SoS LCA allowed for a frank and honest assessment of what areas were most critical to the successful development of FCS software.

In addition, since this was in essence an event to prepare for a PDR-type review, it gave the review team an opportunity to canvas the FCS program to gather in software concerns from all the program IPTs. This allowed even non-software focused groups to offer up what impacts software had on their efforts, and what areas could potentially use more attention in our review.

The result of this effort to canvas the program for software concerns resulted in the development of what the review team termed the “Reference Sheet.” This was an artifact that listed all the gathered software concerns, whether addressed in the SoS LCA effort or not, and noted how those concerns were being reviewed. Many were tagged for review in the SoS LCA, or were being reviewed through a separate effort. However, there were some software concerns that had no review event or other activity examining their impact to the FCS program. Thus, those “homeless” concerns were examined by FCS software IPT management to determine what should be done to ensure they had been addressed, and if necessary, mitigated, prior to the program wide PDR review.

This underscores the wider benefits of the SoS LCA review. It provided a springboard to consider software concerns from all the various perspectives that existed on the FCS program. As a result, the opportunity was presented, and utilized, to develop methods to review all these varying software concerns, whether through the SoS LCA or another activity, and thus having a better understanding of where the FCS software effort as a whole stood entering PDR.

3. Establishing Focus Area Priority

In order to provide a hierarchy to the evaluation and eventual presentation of the data and analyses resulting from the SoS LCA review, it was necessary to establish levels of priority in the focus areas. As such, it was determined that the Software Producibility and End-State Design focus areas would be the two that all other analyses would be rolled-up to support.

Software Producibility

A key phenomenon to be addressed has been the ability of the various software developers to maintain productivity rates across multiple increments. Unlike hardware, where unit costs tend to decrease with added production volume, the unit costs of later software increments tend to increase, due to previous-increment breakage and usage feedback, and due to increased integration and test effort. Thus, using hardware-driven or traditional software-driven estimation methods for later increments would lead to underestimates and overruns in both cost and schedule. Calibrating this phenomenon involves measurement of producibility across early incremental software deliveries.

End-State Design

This effort was perhaps the most ambitious of all the focus areas. The question that initiated it was whether the end state software design (which would be reflective of the software at the conclusion of the FCS software development effort) truly supports the operational needs of the FCS program? The tactic chosen to review this focus area was to perform an assurance case review [SEI, 2007] on each of the FCS Key Performance Parameters (KPPs).

The assurance case model helped the reviewers develop a structured model of claims about the system's properties that would need to be proven true to indicate the KPP could be realized. In order to provide evidence to support the realization (or non-realization) of these KPPs the outputs of the analysis of several of the other event focus areas would be needed. Network Management and Information Assurance, for example, would support a Net-Readiness KPP. Any positive analyses, or concerns as a result of reviewing the lower level focus areas would filter up to the end-state analysis of the ability to reach the needs of the KPP. This same process of analysis is applicable to the realization of the other KPPs.

Analysis of Producibility and End-State Analyses

The results of this end-state design analysis, through the use of data from many of the other focus areas, paired with the software producibility data, allowed for a determination of how much capability could be achieved via FCS software before the Milestone C decision point. The producibility data demonstrated the amount of resources, in terms of people, budget, and schedule that had been utilized to develop the capabilities available on the FCS program to date. Through a comparison of those capabilities against the capabilities planned to be available at this time, an analysis could be performed to determine how many more capabilities could be developed given the time, personnel, and budget in the period from PDR to Milestone C.

Again, this demonstrates the initial goal of the SoS LCA review. It was to provide not only an assessment of the current state of the FCS software effort, but to also provide methods to achieve the most complete software functionality prior to the Milestone C decision point. As the results were rolled-up, it provided the program the proper data to assess what key program events and tests could adequately be performed with the functionality baseline that would indeed be deliverable. Having this data prior to a PDR allows for the development of a realistic design and software production plan that can be accomplished through the development of software capability that will be feasible based on remaining and projected program resources.

4. Results of Data Analysis

The final report from the SoS LCA analysis activities was primarily focused on the Software Producibility and End-State Design areas described in Section 3.

The producibility review included data from all levels of suppliers, including the prime- and sub-contractor level. The results of the analysis were both general (overall analysis and producibility projections for the FCS software development effort as a whole) and specific (looking at specific software developers within the program and the projections for their individual software products). Overall risk ratings were provided as well as individual product risk ratings so that recommendations on risk mitigation could be given at several levels of detail, including small corrections at lower levels and more sweeping changes at a SoS level.

The end-state review leveraged not only its own analysis, but also rolled up the analyses of most of the other focus areas of the SoS LCA review. Through the use of the assurance case methodology, the final report reflected not only any issues or risks with the realization of the FCS Key Performance Parameters (KPPs) as the top level assurance case claims, but also an ability to note any risks at lower levels of the assurance case. This evidence could include individual reports, design, or requirements documents and thus could provide feedback on key information that was missing from those individual documents. This type of analysis, like that of the producibility results, allowed for potential improvements both within an individual system, or at the SoS level. Further, the effects on software producibility of the risks identified in the end-state review were used to refine the estimates of how much software could be produced by Milestone C.

5. Conclusions

While debates over how best to implement functional changes in the FCS program based on the data presented in the SoS LCA report are still occurring at the writing of this paper, in terms of providing an overall process improvement, the SoS LCA concept and event were an unquestioned success. One of the largest benefits of the SoS LCA beyond the analyses regarding the status of the program was that it also served as a proof of concept that an event with a small budget and personnel footprint could produce conclusions that had a very high influence on the path of the overall program. Key to this influence was the timing of the event, as holding it such that it concluded just prior to the program SoS PDR allowed a perfect venue to present and discuss the results. Also incredibly important was management buy-in, not just in terms of providing support and leadership for the SoS LCA process, but also putting completion of the SoS LCA into the contract deliverables to ensure follow-through from the supply contractor(s).

An additional realization from the software producibility portion of the review was that there was no consistency in the amount of data, or quality of data, kept by program subcontractors in terms of software producibility metrics. Each supplier kept track of different sets of data, at different levels of detail, and at different periods of time. Using this variety of data in a compiled manner made the job of drawing conclusions and projections from it even more challenging. Thus, a lesson learned here is that for future efforts, ensuring contract language mandating consistent software metrics in terms of types, frequency, and detail of data can provide untold benefits and ease of analysis to a program. Beyond the consistency of data is also the manner it is presented in at lower-level program reviews. The SoS LCA analysis found that all too often this type of data is relegated to a report appendix rather than being a key factor in the body of a software status report. Increased visibility for this data can allow for better and more continuous review of software producibility metrics, and help in finding any risk areas earlier and at a developer level where they can be more quickly and simply corrected, rather than in a program-wide analysis review such as the SoS LCA.

Similarly, data showing evidence of KPP achievement feasibility should be called out as a contractual deliverable. As such, its planning and preparation activities becomes subject to earned value tracking and monitoring in project reviews. The resulting stronger evidence of feasibility ensures a much sounder basis for proceeding into the successive stages of system definition and development.

The SoS LCA event provided a high-visibility review and analysis of the entire Future Combat Systems software development program, and leveraged that visibility to help implement changes to the program path to ensure capability delivery that benefits the soldier as soon as possible. This success was due to the uniquely broad scope of this software review event, and the unique approach of leveraging data from across software builds and across levels of development to provide a truly unique and valuable set of analyses and recommendations that were implemented by the FCS program to develop more complete and higher quality software to best meet program goals.

6. References

[Boehm, 1996]. Barry Boehm, "Anchoring the Software Process", IEEE Software, July 1996.

[FCS SDP, 2007]. P. Goforth, et al, "Software Development Plan", Future Combat Systems, June 2007.

[FCS SoS LCA Plan, 2008]. A. Nguyen, et al, "Plan System of Systems Life Cycle Architecture Assessment", Future Combat Systems, November 2008.

[FCS SoS LCA Report, 2009]. A. Khan, et al, "Life Cycle Audit Assessment", Future Combat Systems, March 2009.

[SEI, 2007] C. Weinstock, "Assurance Case and Plan Preparation", Software Engineering Institute, June 2007 <http://www.sei.cmu.edu/pcs/acprep.html>.