Software Engineering Institute

Incremental Development in Large-Scale Systems: Finding the Programmatic IEDs

Charles (Bud) Hammons, PhD

June 2009

TECHNICAL NOTE
CMU/SEI-2009-TN-015

Acquisition Support Program
Unlimited distribution subject to the copyright.

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.

Copyright 2010 Carnegie Mellon University.

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

Internal use. Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use. This document may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other external and/or commercial use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about SEI publications, please visit the library on the SEI website (http://www.sei.cmu.edu/library).

Table of Contents

Acknowledgements **iii**

Abstract **v**

1 Introduction **1**

2 Programmatic IEDs **3**

2.1 What Criteria and Processes Will be Used to Determine What is “Operationally Useful?” 3

2.2 CONOPS Varies Across Increments 3

2.3 Traditional Systems Engineering Practices May Not be Entirely Appropriate to the Incremental Development Environment 3

2.4 Unrecognized Software Development Dependencies 4

2.5 Potential for Reuse and Commercial Off-the-Shelf Software May Change 4

2.6 Inter-Increment Dependencies 4

2.7 Unprecedented Software Integration Complexity and Scale 5

References/Bibliography **7**

Acknowledgements

The author wishes to acknowledge the review and contributions of John Foreman, Ron Kohl, Mary Ann Lapham, Tom Merendino, and David Taylor to earlier versions of this document. All are current or former staff members of the Software Engineering Institute.

Abstract

IED (improvised explosive device): p*rogrammatically, an unintended consequence or impediment that can blow up a development program.*

Large-scale systems (LSS) being acquired by the Department of Defense (DoD) are frequently exemplified by the creation of multiple prime items, acquired under separate contract. The multiple prime items are often controlled by different organizations, with attendant variations in timelines and funding stability. In most cases, each of the prime items is software-intensive. LSS are encountered in several domains, including space-based systems and multi-platform systems such as the Army's Future Combat System. These are often referred to as *transformational systems*.

The concepts of time certain development and incremental deployment of capabilities would appear to represent a fundamental change in the programmatic environment in which LSS are acquired. Such programs need a "roadmap" for acquisition that addresses this new environment. This paper explores how continued use of the existing acquisition roadmaps opens up the potential for running into program pitfalls (programmatic IEDs) that aren't acknowledged on the map at hand.

1 Introduction

The concepts of time certain development and incremental development have not yet been fully digested and interpreted for large-scale systems (LSS), but the concepts have the effect of providing a decision hierarchy for program direction, where time is mandated to be a predetermined interval post Milestone A, cost is primarily legislated based on Department of Defense (DoD) recommendations and the action of external entities, and the actual capabilities to be delivered (i.e., requirements) are a matter of some negotiation. It is important to recognize that "time certain" decision criteria and the order in which they are applied are different from traditional models employed on acquisitions to date, which is roughly to address requirements as a whole, with negotiation focused upon cost as an independent variable (CAIV) processes. The upshot is that what is "optimal" using the two different decision models isn't necessarily identical.

At this time, it is not certain that the recommendations of Kadish will be embraced and mandated by the DoD [Kadish 2005]. Regardless, it is important that LSS program teams analyze the effects of such a set of potential mandates. For example, how big a change is this in practical terms? If an LSS development program already in process continues on its current path and the mandates are imposed, what are the downside risks to the LSS? Will there be a grandfather provision for programs already under way? Suppose other major programs follow the mandates and the LSS continues on its current path: does this make the program's relations with the DoD and Congress more problematic? These are business questions that are best addressed by senior leadership.

The rest of this paper addresses programmatic IEDs that an LSS program is likely to encounter.

2 Programmatic IEDs

2.1 What Criteria and Processes Will be Used to Determine What is "Operationally Useful?"

A key proposition in Kadish is to mandate that "Operationally Useful Capabilities" be delivered six years post Milestone A [Kadish 2005]. Even if an LSS program has made extraordinary efforts to engage the user community, it is important to revisit some fundamental questions should the recommendations in Kadish be accepted and enforced by the DoD. Who decides what is "operationally useful?" Who is the authoritative source for information? Has the LSS identified/cultivated a set of users who can "think transformational" and envision the operational value of incremental introduction of LSS capabilities? In Kadish, the strong suggestion is that combatant commanders are the proper source. As LSS programs engage with the users, the user's perspective of what is "useful" will change as their understanding of what can be done operationally becomes more sophisticated. As a result, the concept of operations (CONOPS) and doctrine may change, affecting the requirements to be deployed in a given block or increment. There is clearly an educational role that LSS program personnel can fill to help the user community envision what a force can accomplish with the capabilities that the LSS promises.

2.2 CONOPS Varies Across Increments

Consider a space-based LSS, composed of multiple satellites. That which is operationally feasible after first launch will differ from that after second launch, and so forth. If the constellation is made up of satellites of differing capability, this diversity will strongly influence the details of constellation operations, and may unfortunately have a direct effect on CONOPS. This can thwart efforts to ensure that end users can think in terms of capability delivered to end users without having to address details of constellation operations that are usually the domain of the satellite operators. Differences between satellites may lead to design variations to mitigate the deltas. Some functions that may be carried out in hardware (application-specific integrated circuits and the like) in satellites 2, 3, and so forth may be executed in software in satellite 1. Do we provide additional processor bandwidth and/or memory margin to accommodate the expectation for more software workarounds on the less capable satellite 1? In communications satellite systems, the existence of large inventories of legacy terminals also complicates CONOPS; it is not unusual for full operational capability envisioned to be deferred until the legacy inventory has been replaced.

2.3 Traditional Systems Engineering Practices May Not be Entirely Appropriate to the Incremental Development Environment

Conventional hierarchical program master schedules and plans may not capture the dynamics of the incremental environment. In time certain and incremental development the task is to craft a sequence of increments where increasing capability is delivered to the field, some of which may actually be below threshold levels or absent altogether from the first increment and intermediate upgrades. In our space-based example, the hardware and actual capabilities of distinct satellites may be different. The staging of requirements packages that match user expectations—and can grow from increment to increment—is a non-trivial exercise; it requires that requirements and

configuration and version interdependencies be understood at a deeper level than contained in a static requirements document. Allocation splits between hardware and software may change from increment to increment.

The forces in action also include requirements fluidity, dynamics of the trade spaces being investigated in parallel developmental efforts, and evolution of external systems in parallel development (such as the Global Information Grid). The effect too easily leads to a constant reconstruction of static plans that are invalidated in short order by changes in user expectations, changes in the trade space, and other factors which take management attention away from managing the process of converging on a set of operationally useful capabilities that can be deployed in the existing time and cost constraints.

A common technique to mitigate this problem area is to institute a concurrent development approach. It is customary in such situations to provide a venue and processes to address cross-domain systems engineering issues, particularly the coordination of requirements and interfaces.

2.4 Unrecognized Software Development Dependencies

Software development in each of the prime items of an LSS is dependent on software development efforts development in the other segments. Inter-segment interfaces only describe the nature of the messages exchanged among the segments, but do not ensure that the right thing happens in each segment. This problem is amplified in incremental development, since the negotiation to arrive at what is operationally useful has to span all segments. Configuration control is a particularly difficult issue that spans development and sustainment. The fact that configuration control during sustainment and development is executed by different organizations with goals that are not identical, with sustainment organizations emphasizing modifications to support near-term operational needs, while development organizations emphasize forward-looking capabilities. During development, a greater emphasis on coordination across prime items that comprise the LSS is necessary to create an LSS that meets expectations.

2.5 Potential for Reuse and Commercial Off-the-Shelf Software May Change

The new development environment potentially places additional challenges on the use of commercial off-the-shelf software (COTS) and reusable code. Each increment must provide operationally useful capabilities. Due to the expected evolving nature of the system, and the unknown breakdown of the functionality, what COTS can be leveraged and what algorithms and code may be reused will need to constantly be re-examined. This reveals a potential conflict between the proposed time certain guidance and existing mandates to maximize employment of COTS and reused software.

2.6 Inter-Increment Dependencies

Each increment must be considered an entire system unto itself, requiring the full spectrum of software and systems engineering attention. While this poses little difference from the current plan for Increment 1, it places subsequent increments in a position to treat the previous increments as somewhat of a legacy system that has to be accommodated in these subsequent increment development efforts. This suggests, for example, that a full Increment 1 architectural suite will be required, and then something equivalent will be required for each of the following increments.

This would infer a set of parallel or staggered development efforts, each with its own set of development artifact needs (architectures, requirements, transition plans, etc.). This staggered but nearly in parallel set of development projects, one for each increment, can place unexpected strains on resources that may need to be shared (or not) between various increment development efforts. It is possible that other programs have addressed this issue, and their experience may be useful in formulating a specific strategy for a given LSS program.

2.7 Unprecedented Software Integration Complexity and Scale

In contrast to the other items mentioned, this risk is independent of whether time certain or incremental are chosen as a management approach, or the status quo is retained. Most LSS programs envision the integration of 10^7 or more equivalent lines of code (ELOC) for the total program. We are unaware of any DoD program in the past 20 years that has integrated software of this size and complexity within anywhere close to the originally proposed cost, schedule, and performance parameters. Other notable risks (such as sensor maturity) can be mitigated in part through trade studies, technology maturation, and negotiations of the operationally useful package of capabilities. The integration risk noted cannot be mitigated through any of the mechanisms currently employed on the program. In current plans the effect of this risk will surface in later stage integration and test, a time when the program will again be vulnerable in a very public way. It is important that any LLS program have a vigorous systems integration IPT (integrated product team) empanelled from the beginning of the program, specifically charged to address the downstream integration issues, including software integration. Some acquisition organizations are attempting to adopt an enterprise approach to the various programs in their portfolios. One tactic employed is to institute cross-program configuration control at the requirements level to aid in managing one of the primary drivers of complexity growth. The management of interactions among the prime items in development or in service is also addressed by the institution of test and integrations teams charged with ensuring that dependencies are properly tested and existing operational capabilities are not interrupted.

References/Bibliography

[Kadish 2005]

Kadish, Ron, Lt Gen USAF (RET), et al. *Defense Acquisition Performance Assessment – Executive Summary*. Defense Acquisition Performance Assessment Project, Office of the Acting Under Secretary of Defense; December 2005.
www.defenselink.mil/pubs/pdfs/DAPA%2012-22%20WEB%20Exec%20Summary.pdf

[DoD 2005]

Defense Acquisition Performance Assessment, public meeting, December 14, 2005.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. **AGENCY USE ONLY** (Leave Blank)	2. **REPORT DATE** June 2009	3. **REPORT TYPE AND DATES COVERED** Final
4. **TITLE AND SUBTITLE** Incremental Development in Large-Scale Systems: Finding the Programmatic IEDs		5. **FUNDING NUMBERS** FA8721-05-C-0003
6. **AUTHOR(S)** Charles (Bud) Hammons		
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-TN-015
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**
11. **SUPPLEMENTARY NOTES**		
12A **DISTRIBUTION/AVAILABILITY STATEMENT** Unclassified/Unlimited, DTIC, NTIS		12B **DISTRIBUTION CODE**

13. **ABSTRACT (MAXIMUM 200 WORDS)**

Programmatically, an IED (improvised explosive device) is an unintended consequence or impediment that can blow up a development program.

Large-scale systems (LSS) being acquired by the Department of Defense (DoD) are frequently exemplified by the creation of multiple prime items, acquired under separate contract. The multiple prime items are often controlled by different organizations, with attendant variations in timelines and funding stability. In most cases, each of the prime items is software-intensive. LSS are encountered in several domains, including space-based systems and multi-platform systems such as the Army's Future Combat System. These are often referred to as *transformational systems*.

The concepts of time certain development and incremental deployment of capabilities would appear to represent a fundamental change in the programmatic environment in which LSS are acquired. Such programs need a "roadmap" for acquisition that addresses this new environment. This paper explores how continued use of the existing acquisition roadmaps opens up the potential for running into program pitfalls (programmatic IEDs) that aren't acknowledged on the map at hand..

14. **SUBJECT TERMS** large-scale systems, time certain development, transformational systems, incremental deployment of capabilities, programmatic IED	15. **NUMBER OF PAGES** 16
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