National Defense Industrial Association
Systems Engineering Division
Task Group Report

Top Software Engineering Issues within
Department of Defense and Defense Industry
September 2010

Introduction

The NDIA Systems Engineering Division convened a workshop March 15-16, 2010 to examine the top software engineering issues that face the acquisition and successful deployment of software-intensive systems, and to develop recommendations to address them. The workshop was coordinated with the Director, Systems Engineering, Office of the Under Secretary of Defense, Acquisition Technology & Logistics, who serves as the primary OSD interface to the NDIA Systems Engineering Division.

Members of the task group included 25 participants from 15 organizations across industry, government, and academia (summarized in an attachment to this report). Inputs were solicited in advance, with some organizations providing introductory briefings, including:

- NDIA SE Division 2010 action plan, developed in response to OSD priorities and DDR&E imperatives
- Army Software Systemic Analysis Study
- 2009 NDIA Software Test & Evaluation Summit/Workshop Review.
- Summaries of other key studies and reports from DoD, Congress, and NDIA

The task group considered progress against the Top Software Issues identified in the prior NDIA report from the 2006 NDIA workshop.

<table>
<thead>
<tr>
<th>#</th>
<th>2006 Issue</th>
<th>2010 Status</th>
</tr>
</thead>
</table>
| 1  | The impact of requirements upon software is not consistently quantified and managed in development or sustainment.                                                                                           | • Some progress, but still inconsistent software requirements definition in initial development planning, sustainment.  
• Issues persist with JCIDS capability documents and the resulting definition, management, and flowdown of software requirements.                                                                                                                                  |
| 2  | Fundamental system engineering decisions are made without full participation of software engineering.                                                                                                       | • Improving by necessity in software-intensive systems.  
• Policy updates (5000.2, competitive prototyping) imply software engagement, but are not explicit.                                                                                                                                                                                                                           |
| 3  | Software life-cycle planning and management by acquirers and suppliers is ineffectiveness.                                                                                                                   | • No significant changes. WSARA requirements for independent estimates are not specific for software.                                                                                                                                                                                                                             |
| 4  | The quantity and quality of software engineering expertise is insufficient to meet the demands of government and the defense industry.                                                                  | • Resource issues persist, in government and industry.  
• Shortages in leadership, domain, architects, software engineers  
• Initiatives are in place to improve acquisition workforce, STEM, cross-training                                                                                                                                                                                                                                           |
Traditional software verification techniques are costly and ineffective for dealing with the scale and complexity of modern systems.

Better focus (e.g., SW T&E Summit), but issues remain
Greater testing emphasis in OSD policy and organization
State of the practice techniques are not widely utilized

There is a failure to assure correct, predictable, safe, secure execution of complex software in distributed environments.

Much progress due to emphasis (e.g., intrusion), but issues persist.
Need to implement best practices (System Assurance Guidebook)

Inadequate attention is given to total lifecycle issues for COTS/NDI impacts on lifecycle cost and risk.

Open source: must consider data rights, IP issues.
Reuse: opportunistic; some repositories, resources.
COTS: address obsolescence issues.

Identification of Top Software Issues

The task group identified and recorded numerous root issues in initial discussions, which were classified into affinity groups for discussion and refinement. Several common themes regarding the current acquisition and development environment became evident:

- The need for rapid development has increased further; we must find ways to be more responsive to urgent mission needs (faster, better, cheaper).
- The need for better software architectures and modular components has been reinforced, and can be more difficult when accounting for legacy upgrades, systems of systems (SoS), and interoperability.
- Stakeholder commitment to agile development processes is not widely understood – agile terminology is often misused (e.g., acquisition agility is not the same as Scrum methodology).
- The software community must be engaged in advanced development planning.

These themes helped the task group to characterize the areas where action is needed:

- **Rapid acquisition** – responding to urgent mission needs (faster, better, cheaper).
- **Starting programs right** - effective program planning and start-up.
- **Executing programs right** - consistently implementing and managing programs in accordance with best practices, while identifying areas where current practices may not be adequate to address the needs of future defense systems.
- **Sustainment** – inadequate consideration of total life cycle activities.

Sub-groups were formed to expand and refine these concepts into the NDIA Top Software Issues of 2010:

1. **Development of software and systems must occur more rapidly than current acquisition models and development processes allow in order to meet the increasing demands of the warfighter.**
2. **Inadequate/insufficient program planning, start-up, and management of baseline changes has a negative impact on the program’s SW success.**
3. **Current software development and management practices have not kept pace with emerging needs in areas such as complex systems, Systems of Systems, system assurance, software verification, and COTS/NDI integration.**
4. **There is insufficient attention given to the overall software life cycle activities including sustainment and changing threats.**

Details of these issues and recommendations to address them, developed through task group consensus, are described in the following sections.

**Issue 1:**

*Development of software and systems must occur more rapidly than current acquisition models and development processes allow in order to meet the increasing demands of the warfighter.*
The following main points provide amplification of this issue:

- **Acquisition and development processes must be more responsive to urgent mission needs to deliver capabilities to the field more quickly.** The current DoD acquisition environment is not well-suited to address these needs, and agility can be encumbered by mismatches in processes, terminology, milestones, and management methods.
- **Prioritization of capabilities within an aggressive deployment schedule requires cooperative engagement between the acquirer, developer, and warfighter.**
- **Rapid development and deployment of new capabilities can be accelerated through reuse of existing systems, and use of open source and COTS components.** This depends heavily on software architectures that are well-designed to flexibly accommodate rapid development, change, test, verification, and fielding of new capabilities, as well as effective sustainment capabilities to quickly upgrade and maintain systems.
- **Rapid development and adaptation of systems requires a multi-disciplinary approach, yet this is not often emphasized outside of the software environment.** Cross-disciplinary processes are not generally in place to responsively provide the concurrent engineering and management approaches needed.
- **For rapid development and deployment, there are often inherent tradeoffs that must be managed between an appropriate level of engineering rigor during development and the sustainment issues encountered post-deployment.**
- **Prototypes are often declared operational as soon as they are introduced to the warfighter without comprehensive life cycle engineering and adequate sustainment infrastructure.** Prototypes leveraging existing systems can also be inhibited by issues such as organizational accountability, configuration management and sustainment.

**Recommendations:**

**NDIA to facilitate a study or working group to characterize best practices and determine the measured effectiveness of rapid deployment techniques.**

- Identify critical elements in design and implementation processes to achieve a successful rapid development and deployment result. Clearly differentiate the value of various techniques that enable or inhibit rapid development.
- Create guidelines to promote an environment that enhances appropriate use of agile software development processes and terminology, including those for software estimation and measurement, and to characterize the similarities and differences relative to rapid acquisition.
- Identify tools and techniques that enable multi-disciplinary collaboration (e.g., software and systems engineering) in support of rapid development (i.e., to eliminate discipline-specific stovepipes).
- Establish guidelines for rapid development and deployment environments, including the application of reuse, open source, and COTS/GOTS components.

**NDIA to establish a joint industry/government working group to investigate new acquisition approaches that accommodate rapid development.**

- Engage key government stakeholders (e.g., Rapid Fielding Office, Defense Portfolio Manager) in providing guidance to acquirers to fund and support interactive interchanges that enable rapid decisions and that gain the benefits of agile software development processes.
- Provide contractual language and guidance that facilitates acquiring software for rapid system deployment. Examples of items to be addressed include specifying flexible software architecture attributes and enabling prioritization of the capabilities and requirements that support rapid development, updates, and sustainment.

**Issue 2:**

**Inadequate/insufficient program planning, start-up, and management of baseline changes has a negative impact on the program’s SW success.**

The following main points provide amplification of this issue:

- **The emphasis on advanced development planning in the acquisition process means that software architecture and trade-offs for achieving system performance capabilities through software must play a larger and more significant role much earlier than in past development programs.** Our current software engineering
methodologies and practices may not be well suited to provide the knowledge and tools necessary to achieve this goal.

- Many software programs start execution without a clear and stable understanding of requirements and the criteria for successful completion (e.g., Undefinitized Contract Action (UCA)). Changes to scope and expectations are not always reflected in accurate plans, budgets, and schedules, which can ultimately result in not meeting commitments and not delivering the full functionality expected.
- When certain software functions are retrofitted late in a program instead of planned up front they may create significant – possibly prohibitive – cost and schedule expansion. Examples include encryption on data downlinks, data recovery, error handling, information assurance, scalability factors, and numerous other problems.
- Explicit software engineering plans are not consistently required in RFPs, and may not be adequately addressed in the program plan or Systems Engineering Plan. Integrated master plans should reflect a shared inter-disciplinary understanding and agreement of software requirements, with dependencies, interfaces, and commitments reflected in plans, schedules, budgets, risks, and milestones with quantitative criteria.
- Software expertise is not always engaged adequately in early planning and system architecture development in a way that would ensure consideration of software aspects in program plans, technologies, trades, and risks/contingencies. Decision-makers may not be well-informed on software-related impacts. This can often result in success-oriented plans and capabilities that are not realistically achievable, or that require significant rework later in the life cycle where these oversights are more costly to correct.
- Software estimation and measurement practices are not consistently applied and therefore their effectiveness is limited. There is often a disconnect between the measurement processes used for software estimation and those used for software work planning and monitoring progress.
- Software plans and adequate resources are not consistently adjusted to accommodate changes to requirements, threats, priorities, and emergent or obsolescent technologies and environments. Requirements are not consistently well-defined, validated, and limited to what is achievable within programmatic constraints; this can result in unachievable plans and poor program execution.
- Programs (including government and contractor) are often not well-structured to accommodate software changes in the program life cycle. Areas where organizational issues are commonly encountered include roles/responsibilities, silos, risk contingency reserve, test approaches, integrated master schedule, funding type and timing.
- Contracts are not always well-structured to ensure open dialog with bidders, especially in a competitive environment. Contracts often do not easily accommodate evolution or change to mission needs, requirements, acquisition strategies or funding levels.
- Contract terms and conditions are not consistently flowed down adequately to subcontractors with adequate oversight, including quantitative criteria for quality factors (including reliability), interfaces, and quality assurance.

**Recommendations:**

**NDIA to provide best practices, in government and industry, that encourage the use of program assist start-up teams working directly with programs to develop comprehensive program plans, processes, and organizational structures.**

- Startup assist teams are effectively used in some companies and programs to help programs through the critical initial program planning and startup period. Gather and publicize concepts and best practices from industry/government resources (e.g., NDIA Industrial Committee on Program Management (ICPM), DoD transition workshops for program startup, DAU startup process with role definitions, industry experience/reports). Validate startup guidance with research and evidence-based reasoning based on actual program successes. Consider models such as a “teaching hospital” or “green team”.

**NDIA to develop guidance and assets for improving planning and organization on software-intensive programs, with emphasis on improving the integration of software engineering with systems engineering and other program functions.**

- Create a handbook on planning and organizational structures for software-intensive programs, including the explicit roles, responsibilities, accountability and inter-disciplinary interfaces for software engineering. This could include, for example, government and contractor lead software engineers, and descriptions of their engagement with other functions or IPTs, such as systems engineering and the program integration team.
• Provide recommendations on aligning and integrating software plans and documentation with those from other program functions (e.g., systems engineering, product architecture, drawing trees, IMP/IMS, risk mitigation plans, quantifiable measures), with event-driven criteria for program milestones, work product maturity, and technical reviews that ensure software issues are adequately addressed across the program life cycle. This should include a systematic assessment of all software policies, standards, and guidance documents for software acquisition, engineering, and management to ensure relevance to current and future technology and practice as well as consistency across these controlling documents.

• Provide recommendations to update government policy, guidance, and assets (e.g., RFPs, SEP DIDs) to adequately integrate software as an integral part of the program team, system architecture, and trade space. Recommend establishment of a human capital policy to ensure adequate skills, training, and resources for software engineers and managers, in both government and industry.

Issue 3:

Current software development and management practices have not kept pace with emerging needs in areas such as complex systems, Systems of Systems, system assurance, software verification, and COTS/NDI integration.

The workgroup noted that basic software management practices for traditional development systems are generally well understood, yet these best practices are not always consistently followed. These issues and underlying root causes are well-documented in numerous other studies and reports. The NDIA/DoD Systemic Root Cause Analysis (SRCA) study and report (now continued by the U.S. Army) is a good example of analyzing the underlying root causes for performance problems observed on defense software programs.

Many of these issues have persisted for decades, and are being addressed by new government policies and legislation. Therefore, although these issues are not yet solved, the workgroup did not see it necessary to continue to repeat them here in this report. Rather, the workgroup considered what software management issues are likely to face defense programs in the future as mission needs, threats, and technologies continue to evolve. The workgroup anticipates that future needs will require improved capabilities to effectively manage systems of increasing size and complexity, and to deal with critical threats such as software assurance.

The following main points provide amplification of this issue:

• Current software engineering methodologies and common practices do not scale well to deal with the continued growth in scale and complexity of defense systems. Research into complex systems and systems of systems (SoS) is still emerging, but practices are not yet mature and well-defined enough to be consistently implemented effectively. Software engineering practices and architectures may need to evolve to accommodate rapid changes in technology, development paradigms, and emerging mission needs.

• Current software measures and indicators do not provide adequate insight into these new types of development programs and technologies.

• Traditional software integration and verification techniques are costly and ineffective for dealing with the scale and complexity of modern systems. Software integration strategies are often not well-integrated with system-level approaches.

• The state of the practice for system assurance on defense programs is not adequate to anticipate and mitigate emerging vulnerabilities and threats.

• Plans, estimates, and assumptions for integration and use of non-development items (COTS, GOTS) are often overly optimistic. This can result in cost/schedule impacts and failures to adequately address risks in the supply chain. COTS products often do not deliver the capabilities expected, and may require more integration effort and development of middleware than planned.

Recommendations:

Government fund ongoing research into scaling software engineering practices and measures for complex systems and systems-of-systems.

• Develop/enhance guidance on comparison and selection of appropriate methodologies and architectures for software engineering of complex systems and systems of systems. Build on research conducted at academic institutions such as USC and SE UARC.
• Provide process definition models (for example SE UARC process decision table).
• Sponsor research into appropriate software engineering measures for complex systems and SoS, such as that conducted at the NSF Software Engineering Research Center. Sponsor workshops, in conjunction with leading measurement groups such as Practical Systems and Software Measurement (PSM), dedicated to addressing measures for complex systems and SoS.

**NDIA and government collaborate on enhanced guidance for software integration and verification, and strengthened approaches for ensuring adequate software quality practices on defense programs.**

• Sponsor work groups and other mechanisms to identify the state of the practice in software verification strategies, techniques, and measures.
• Define policy, guidance, and criteria for implementation of software verification techniques, such as modeling and simulation, static analysis, and automated regression testing.
• Enhance policies and guidance for including adequate integration plans and strategies in program planning (SEP, SEMP, IMP/IMS, architectural partitioning, reviews).
• Develop policy and guidance for standard software quality measures that should be collected and analyzed on defense programs throughout the software life cycle.

**NDIA and government define and communicate best practices for system assurance, and implement mechanisms to ensure their use on defense programs.**

• Enhance guidance, such as in System Assurance Guidebook, to address engineering for resiliency in the face of evolving threats. This guidance should also inform the acquisition and development workforce of publicly available resources for identifying and mitigating common vulnerabilities and weaknesses.
• Conduct benchmarking of best practices and research for system assurance, in government, industry, and academia.
• Implement government policies and guidance to improve the treatment of cybersecurity and other assurance concerns in system requirements baselines and architecture definitions to manage vulnerabilities in defense programs.
• Provide training and courseware for system assurance, in both government and industry, to improve awareness of vulnerabilities, implement secure coding practices, and ensure proper use of development and analysis tools.

**NDIA to develop guidance on strategies and practices for COTS/NDI usage and verification.**

• Collect and publish best practices to manage COTS/NDI dependencies and supply chain risk, including COTS/NDI constraints on system requirements, architecture, and implementation.
• Provide guidance on verifying COTS/NDI capabilities in the intended system environment.

**Issue 4:**

**There is insufficient attention given to the overall software life cycle activities including sustainment and changing threats.**

The following main points provide amplification of this issue:

• Most modern weapon systems are extensively dependent on software-driven functionality, but sustainment resources are unable to keep up with changing mission scenarios.
• Sustainment activities for complex SW-driven weapon systems often drive a majority of a program’s life cycle costs, but sustainment is not always considered adequately in system life cycle planning, engineering, and budgets.
• Mechanisms for sustainability are not always built into the software architecture and design.
• Emphasis on competitive prototyping may negatively impact the consideration of sustainability and life cycle cost issues in system level trades.
• Mandated boundaries in funding, roles, and organizations make it difficult to get focused attention on sustainment issues.
• Government life cycle cost estimates developed early in materiel concept trades constrain the quality of system baseline sustainment.
Recommendations:

Government to promote a product line focus within program offices to bring emphasis to sustainment issues (e.g., training, guidance).
  • Emphasize sustainment issues by elevating their organizational visibility within program offices, and by using product line practices to obtain greater alignment, commonality and leverage of sustainment resources across related programs.

Establish a government/industry task group to consider “color of money” issues and the flexibility that does exist to help mitigate the impact on sustainability.
  • Consider innovative ways to make more effective use of sustainment funding and resources,
  • Evaluate contract types, award fee structures, and other incentives for development programs that encourage long-term thinking and sustainability.

Convene a joint government/industry workshop (including user community) to consider software sustainment issues across the program life cycle.
Summary
The Task Group appreciates this opportunity to provide input on the greatest issues facing the defense software industry. The Task Group believes action upon these issues and recommendations, by both OSD and industry, will have substantial positive impact upon the planning, acquisition, and execution of DoD software-intensive programs. The NDIA Systems Engineering Division looks forward to continued collaboration with OSD in addressing these solutions.

Summary Recommendations

Rapid Acquisition:
1. NDIA to facilitate a study or working group to characterize best practices and determine the measured effectiveness of rapid deployment techniques.
2. NDIA to establish a joint industry/government working group to investigate new acquisition approaches that accommodate rapid development.

Starting Programs Right:
3. NDIA to provide best practices, in government and industry, that encourage the use of program assist start-up teams working directly with programs to develop comprehensive program plans, processes, and organizational structures.
4. NDIA to develop guidance and assets for improving planning and organization on software-intensive programs, with emphasis on improving the integration of software engineering with systems engineering and other program functions.

Executing Programs Right:
5. Government fund ongoing research into scaling software engineering practices and measures for complex systems and systems-of-systems.
6. NDIA and government collaborate on enhanced guidance for software integration and verification, and strengthened approaches for ensuring adequate software quality practices on defense programs.
7. NDIA and government define and communicate best practices for system assurance, and implement mechanisms to ensure their use on defense programs.
8. NDIA to develop guidance on strategies and practices for COTS/NDI usage and verification.

Sustainment:
9. Government to promote a product line focus within program offices to bring emphasis to sustainment issues (e.g., training, guidance).
10. Establish a government/industry task group to consider “color of money” issues and the flexibility that does exist to help mitigate the impact on sustainability.
11. Convene a joint government/industry workshop (including user community) to consider software sustainment issues across the program life cycle.

For additional information on this Report or other NDIA Systems Engineering Division activities, please contact Bob Rassa, SE Division Chair, Raytheon, RCRassa@Raytheon.com, or Hal Wilson, SE Division Vice-Chair, Northrop-Grumman, Hal.wilson@ngc.com, or Geoff Draper, SE Division 2nd Vice-Chair, Harris Corp, GDraper@harris.com, or alternately, the NDIA Committee Executive, Mr. Sam Campagna, scampagna@ndia.org
Appendix A
Study Participants

The following organizations participated in the generation of the study, by providing input for consideration and/or generating the actual study report.

The Boeing Company
Booz Allen Hamilton
CSC
Harris Corporation
Lockheed Martin
NASA
Northrop Grumman
Office of the Undersecretary of Defense for Acquisition, Technology and Logistics (OUSD (AT&L), DDR&E)
Raytheon Company
Rockwell Collins
Software Engineering Institute
Stevens Institute
U.S. Army
U.S. Air Force Office of Scientific Research (AFOSR)
U.S. Air Force