Software Sustainment

Defense Acquisition University
Jason Hamilton
Professor, Information Technology
7 June 2017
Why does Software Sustainment Matter?

• Most DoD systems are software reliant.
  • Maintenance and sustainment activities are disproportionately increasing.
• Operations and sustainment costs generally comprise 70-90% of the total lifecycle costs.

Source: CARD data, Federal Procurement Database System, QSM, CSIS Analysis
The Software Iceberg: More to Manage than Meets the Eye

Must plan to manage more than just this

- Many considerations when conducting software sustainment planning.

- Generally, the software development environment needs to be replicated for sustainment.
Software Sustainment Key Considerations: Sustainment vs Maintenance *definitions*

- **Software Sustainment**
  - Comprehensive requirements to support, maintain and operate the software capabilities of a system.
  - Includes processes, procedures, people, material and information required to support, maintain and operate the software aspects of a system (SEI definition).

- **Software Maintenance**
  - The process of modifying a software system after delivery to correct faults, improve performance or adapt it to a changed environment (IEEE definition)
  - *Actual changes/updates to the software code.* One facet of software sustainment.
  - Essentially development, *creating a new version (baseline).* (*This is a fundamental difference from hardware maintenance*)

- **Implications**
  - Software maintenance drives additional considerations inherent to a new version (baseline), including configuration management/control, updated manuals and training, distribution process (release management), etc.
Software Support Planning and the Acquisition Life Cycle

- SSA options

Legend: ▲ - Milestone Decision ○ - Decision Point

*The actual number and type of builds during the program will depend on system type.
Software Sustainment Planning; Know your Cost Drivers

• There are several considerations during software development that can impact sustainment costs.
  • Requirements volatility
    • Evolving requirements after delivery results in additional cost
  • Poor developer process maturity
    • Poor maturity can result in quality deficiencies reaching the end user
  • Lack of software documentation
    • Poor documentation results in extra troubleshooting time
  • Lack of configuration management
  • Complexity of system software
    • Highly complex software tends to be more unstable and difficult
  • Commercial software licensing implications
    • Considerations such as Enterprise license, per seat license, etc.
Software Support Activity (SSA)

• Assumes the role of providing post-deployment software support for modifications or upgrades made to a system's software following the system's initial fielding.

• The SSA organization typically compiles updates into formal software releases to avoid disrupting the fielded system.

• Software maintenance activities performed by a SSA are the same as those carried out during the development effort that led to the first fielding. They are tailored, as appropriate, to reflect the effort required to implement each change package, update pertinent documentation, verify the changes, and distribute the changes to users.
Software Support Activity (SSA), cont.

- Due to the scope and nature of SSA activities, *they should be involved early.*
- Considerations/Lessons Learned include:
  - Identifying the sustainment strategy early: What functionality/authority will the SSA have, and what information will they require?
  - Use an early PS BCA to identify candidate SSAs
  - Working with the SSA to identify their needs, ensure appropriate development contracts/deliverables are in place.
    - *This includes the identification of data rights!*
  - Establish SSA performance expectations – measures/metrics
  - Use SLA/MOA/MOU to articulate the SSA relationship.
  - SSA can help create the Software Transition Plan (STrP) to discuss how they will assume control from the developer.
Software Transition: Criteria

Necessary activities prior to transitioning to sustainment:

• Source of Repair Assignment Process (SORAP)
• Test & Evaluation
• Stable Software Baseline
• Complete Documentation
• Authority to Operate
• Software Transition Plan
• Staffing and Training Plan (MER/MP&TP)

The Software Transition Plan identifies the resources needed to support delivered software and describes the developer’s plans for transitioning delivered software to the support agency.
Like hardware, new software typically has a fairly high failure rate until the bugs are worked out. Once the initial bugs are resolved, failure rates are dramatically reduced. Theoretically, software failures would stay at that low level, identified by the 'infant mortality' failure curve. However, once in O&S (during PDSS) it undergoes recursive patching to resolve latent defects, increase security, or insert additional user requirements. These changes effectively ‘reintroduce’ infant mortality as the code is continually modified.
Software Sustainment Key Considerations: Types of Software Maintenance

• Corrective Maintenance
  • Reactive modification to correct discovered problems

• Adaptive Maintenance
  • Modification to maintain usability in a changed environment

• Perfective Maintenance
  • Provide functional enhancement to users

• Preventative Maintenance
  • Enhance maintainability, (e.g. ‘refactoring’)

![Diagram showing the types of software maintenance and their percentages.]

Types of Software Maintenance

- Preventative, 5%
- Corrective, 20%
- Perfective, 50%
- Adaptive, 25%

Perfective, Adaptive, and Preventative (80%) are considered “software evolution”
Software maintenance activities include:

- Understanding Requirement
- Understanding Existing Code
- Implementing Change
- Checkout & Deployment

However, poor documentation of existing code can increase the effort required to ‘understand the product’ – up to 50% of the entire sustainment effort. The effect:

- Increased sustainment costs,
- Longer schedule requirements, or
- Reduced testing and debugging (= higher failure rates post-deployment).

"Pay me now or pay me later..."
IT Disposal: Considerations

• **Hardware**
  • Consider residual data that can be reconstructed – sanitize!

• **Storage Media**
  • Consider residual magnetic, optical electrical or other data representations – sanitize!

• **Software (must also consider data remanance)**
  • Overwriting
  • Degaussing
  • Destruction

(Source: IPSE Guidebook, 2011; Computer Resources Support section)
(Some) Key Documentation

• Software Development Plan
  • How the developer is building the software, including software assurance (secure coding)

• Software Transition Plan
  • How the SSA will assume responsibility of the effort from the developer

• Life Cycle Sustainment Plan
  • Section 1; summarize use of COTS, licenses, data rights, cost drivers etc.
  • Section 2; identify the sustainment performance measures – vendor and SSA
  • Section 3; provide details about SSAs role and responsibility, specific licenses, help desk procedures/trouble resolution process, and the agreements that articulate them
  • Section 7; itemize the cost drivers, and offer plans for how to influence them.
  • Section 9; discuss the analytical methods that were used to optimize sustainment (e.g. defect tracking). What tools and processes will be used to track performance and gather user feedback?
Resources

- DAU CLMs
  - CLL056 Sustainment of Software Intensive Systems
  - CLE041 Software Reuse
  - CLE060 Practical Software and Systems Measurement
  - CLE063 CMMI
  - CLE068 Intellectual Property and Data Rights

- DAU Courses
  - ISA101/201/301/320
Summary

• Software sustainment planning should begin early in the lifecycle
  • Identify and involve an SSA
  • Get the appropriate data rights (before MS B)!

• Software maintenance creates a new baseline/version

• Understand your cost drivers, and think about how to mitigate them.

• There are resources available at DAU.mil that can help…
Questions?
Backups
COTS–Intensive Systems — Impact on Sustainment (1 of 2)

• **System obsolescence, technology refresh, and upgrade planning**
  • Each COTS software product life cycle includes updates, refreshes, and obsolescence.
  • Life cycle is not based on the users' requests or budgetary cycles, but rather on marketplace demands and COTS software vendors’ business plans.

• **Source code escrow**
  • Source code may be owned by the COTS vendor or the third-party integrator.
  • Problems can arise when the COTS vendor goes out of business or no longer exists due to a business merger or acquisition.

• **Vendor license management**
  • During development, licenses may be managed by the system integrator.
  • The transition of license management tasks to the sustainment organization needs to be jointly planned by the program office and sustainment organization.

• **Architecture and COTS software interfaces**
  • During system development, third-party integrators/developers may capitalize on relationships with COTS software vendors to acquire system-specific capabilities.
  • These capabilities may not be in the official version of the product and there is no guarantee that these “extra” features will be maintained as the product evolves.
• Choosing a government SSA/Product Support Provider
  • Data rights and licensing considerations with COTS items
  • Service Level Agreements/MOU/MOA – (GOTS?)
  • Functional authority (Help Desk, Configuration Management/Change Control, Information Assurance, etc)
  • Performance measures reporting/frequency

• ‘Modified’ COTS
  • Discouraged, creates unique configurations that marginalize COTS benefits.

• Counterfeit/Grey Market/Malicious software considerations
  • DODI 4140.67 Counterfeit Prevention Policy, Apr 2013
    • CLL032/CLL062
    • DODI 5200.39 Critical Program Information Protection within DOD

• Warranty considerations
  • BCA to consider ROI for extended warranties