

COTS-Based Systems:

COTS Software Lessons Learned, Recommendations and Conclusion

**Richard J. Adams
Suellen Eslinger**

© 2000 Copyright, The Aerospace Corporation, All Rights Reserved

Outline

- **Study Purpose and Approach**
- **Lessons Learned**
- **Recommendations**
- **Conclusion**

COTS-Based System Study

- **Definition:** A COTS-based system (CBS) is a system that contains commercial-off-the-shelf *software* components as elements of the system.
- **Study Purpose:**
 - Synthesize and share lessons learned from actual CBS development and sustainment experiences on SMC and NRO programs
 - Provide recommendations to help mitigate the risks inherent in CBS development and sustainment

CBS Study Approach

- **Step 1**
 - Developed a COTS software experience questionnaire
 - Interviewed SMC/NRO program representatives using the questionnaire as a discussion framework
 - Obtained any available material on COTS software experiences and lessons learned
- **Step 2**
 - Consolidated the results into one experience list
 - Performed an iterative series of analysis and synthesis to identify significant lessons learned
- **Step 3**
 - Developed recommendations to help mitigate the risks inherent in CBS development and sustainment

COTS Software Capabilities are Essential!



- **Today's complex military systems require the leverage provided by COTS software.**
- **COTS software enables focusing on meeting the military's unique needs.**

Lesson Learned - 1

Critical aspects of CBS development and sustainment are out of the control of the customer, developer and user.

- **Vendors are market driven.**
 - The military is not the market.
 - The market may diverge from military needs.
- **Vendors' strategy and market position may change.**
 - Go out of business
 - Drop or de-emphasize products or platforms
 - Change the type and quality of customer support
 - Change or drop promised features, performance and updates

Lesson Learned - 1 (Cont.)

- **Product upgrade quality and content are unpredictable.**
 - Focus on features to attract new customers
 - Focus on fixes for primary customers
 - Limited testing, including regression testing
 - Features added but performance degraded
 - Computer resource requirements increased
 - Incompatibility with other products introduced
 - Backward compatibility with earlier versions eliminated
- **Product upgrade schedule may:**
 - Be time-to-market driven (frequency and release dates).
 - Slip features and fixes.
 - Depend on upgrades to other COTS software.
- **Product and service costs are market driven.**
 - Fees and fee structures may change (licenses and services).

Lesson Learned - 2

Full application of system and software engineering is required throughout the CBS life cycle.

- **Systems and software engineering are still required.**
 - Incorporating upgrades requires a full development effort.
 - Conflicts among multiple COTS products add complexity.
 - Using COTS software only shortens part of the software life cycle.
- **The CBS architecture must support COTS software evolution/replacement.**
 - True “plug and play” capability does not exist.
 - Computer resource margin and growth path must be sufficient.
- **Hands-on prototyping in a system context is essential.**
 - Integration of COTS software can cause unexpected impacts.
 - e.g., performance degradation, product incompatibilities

Lesson Learned - 2 (Cont.)

- **Safety, security and supportability must be designed into the CBS.**
 - COTS software focuses on the commercial marketplace.
 - COTS software is designed independently, not as part of a system.
 - CBS vulnerabilities can be determined by the products used.
- **Product evolution requires continual evaluation of COTS software.**
 - Use multi-dimensional evaluation criteria, not just functionality.
 - e.g., unique military needs, legacy interfaces, vendor characteristics, operations concept, cost
 - Evaluate computer hardware and COTS software together.
 - Prepare backup strategies and contingency plans.

Lesson Learned - 3

CBS development and sustainment require a close, continuous and active partnership among the customer, developer and user.

- **The customer, developer and user must be prepared to trade cost, schedule, performance and O&M concepts.**
 - Must prioritize requirements initially and reassess as necessary
 - Must understand which requirements can be relaxed to achieve a COTS-based solution
 - May need to re-engineer O&M procedures to accommodate a COTS-based solution
 - May discover COTS limitations and incompatibilities at any point in the life cycle

Lesson Learned - 3 (Cont.)

- **Customer, developer and user must be active partners to ensure:**
 - Adequacy of major trade decisions.
 - Full understanding of the evolving CBS capabilities.
 - Acceptability of delivered CBS.

Lesson Learned - 4

Every CBS requires continuous evolution throughout development and sustainment.

- **Currency with COTS software upgrades is essential.**
 - Delaying upgrades exacerbates system impacts.
 - Vendors support only a limited number of past releases.
 - Technology turnover occurs every 12 to 18 months.
 - Hardware upgrades may require COTS software upgrades.
- **External organizations or systems can drive COTS software upgrades, replacements or additions.**
 - DII/COE changes
 - GOTS changes
 - Legacy military system interfaces

Lesson Learned - 4 (Cont.)

- **COTS software may need to be replaced or added at any time.**
 - Elimination of product support by vendor
 - Divergence of product from system needs
 - Increased costs for licenses or support services
 - Identification of unacceptable limitations or vulnerabilities
 - Changes in functionality or performance
 - New or modified user needs
- **Modifying COTS software should be a last resort.**
 - Constrains the CBS evolution path
 - Requires a long-term relationship with COTS vendor
 - Increases life cycle costs

Lesson Learned - 5

Current processes must be adapted for CBS acquisition, development and sustainment.

- **System and software engineering processes must:**
 - Provide robust COTS software evaluation and selection criteria.
 - Require iterative life cycle models with extensive prototyping.
 - Integrate top-down and bottom-up development methodologies.
 - Require incorporation of COTS upgrades during development.
 - Account for COTS software in safety, security and supportability.
 - Enhance configuration management for COTS software complexity.
- **Time and effort need to be reallocated.**
 - More to evaluation, prototyping and analysis
 - Less to software implementation
 - More to integration

Lesson Learned - 5 (Cont.)

- **Customer and user processes must:**
 - Mandate prioritization of user requirements.
 - Result in contracts compatible with contractor CBS processes.
 - Result in milestones compatible with CBS development schedules.
 - Allow flexible and efficient response to unexpected impacts.
 - Support the schedule variability of COTS software upgrades.
 - Provide standardized user safety certification and security accreditation.
- **Standardized licensing processes are needed:**
 - To support maintaining COTS software license currency.
 - To ensure suitability for military needs.
 - e.g., no expiring keys, no export restrictions

Lesson Learned - 6

True cost and schedule of CBS development and sustainment are underestimated.

- **Overlooked or underestimated tasks**
 - Systems and software engineering
 - Hands-on prototyping
 - Integrated system training and documentation
 - Acquisition of COTS software in-depth knowledge
 - e.g., mentors, toolsmiths, vendor support
 - Component and system performance tests
 - Component and system regression tests with each upgrade
 - Related software changes to support COTS software upgrades
 - Glue code, database changes, configuration files
 - Developer/operator training needed for COTS upgrades

Lesson Learned - 6 (Cont.)

- **Unexpected impacts**

- Changes to license or service fees
- Conflicts with the vendor's market
 - e.g., vendor charges to fix problems, refuses to support upgrade/platform, charges for escrowing source code.
- Identification of COTS software limitations or problems (possibly with each upgrade)
 - e.g., performance degradation, interface changes, version incompatibility, new bugs, increased computer resource usage, insufficient documentation, amount and complexity of glue code, need for additional newly developed software
- Externally caused COTS software upgrades/replacements
- Problems or incompatibilities discovered during integration
- Interdependencies of COTS software upgrades

Recommendations

The Government needs to be an intelligent CBS buyer.

- **Prepare for inherent uncertainty in CBS development and sustainment.**
 - Plan for cost and schedule management reserves for COTS software-related problems
 - Plan and contract for:
 - Upgrade strategy during development and sustainment
 - Balanced solution among COTS, reuse and new development to meet cost, schedule and performance objectives
 - IPPD to ensure a close, continuous and active partnership among the customer, developer and user
 - Full life cycle systems and software engineering
 - Additional emphasis on safety, security and supportability

Recommendations (Cont.)

- **Adapt Government processes to support CBS development and sustainment.**
 - Requirements prioritization in partnership with the user
 - Flexible and efficient responses to unexpected impacts
 - Close, continuous and active partnership with developer and user
 - Cost, schedule and performance trades at any point in time
- **Establish horizontal engineering initiatives to support CBS development and sustainment, for example:**
 - Repository for actual experiences with COTS software products
 - Guidance for CBS life cycle cost and schedule estimation
 - Specific acquisition guidance
 - e.g., recommended contract structure and language, use of evolutionary acquisition

Conclusion

CBS benefits are achieved only by careful preparation and execution!

