Software Challenges and Keys to Success

Sept 2016

Joe Heil

Lead of the Naval Software Community of Practice (SW COP)
Lead of the Naval Syscom System Engineering Stakeholders Group Software Working Group (SESG SW-WG)
Chief Engineer and Principal Software Engineer for the Strategic and Computing Systems Dept
Naval Surface Warfare Center Dahlgren Division (NSWCDD)

540-653-1937
joe.heil@navy.mil
Presenter Bio & Presentation Sources

Presenter: Joe Heil

- Over 30 years of applied Naval Warfare Systems Software Development and Leadership
- 20 years as a SW engineer for the Tactical Tomahawk Missile Weapon Control System (TTWCS)
  - Software Developer, Group Lead, Branch Head, and Software Integrated Product Team (IPT) Lead
- **Current software leadership responsibilities:**
  - Chief & Principal Software Engineer for the NSWCDD* Strategic and Computing Systems Dept
  - Lead: Naval Software Community of Practice (SW COP)
  - Lead: Naval System Engineering Stakeholder Group (SESG) Software Working Group (SW-WG)
- **Primary Goal:** Improving naval software system acquisition and engineering success via increased awareness and application of best software engineering practices

Presentation Information Sources

- **First hand experience leading complex software intensive warfare system development efforts**
- **Over last few years:** Chaired/Panel-Member for 300+ Warfare System Project Reviews
  - Wide range of development approaches (Strategic, Waterfall, Agile, Rapid, Prototyping, etc)
  - Wide range of Warfare Systems (Missiles, Guns, Lasers, Directed Energy, Non-lethal, Simulations, etc)
- **Numerous Studies & Reports**
  - Defense Science Board, Gov’t Accounting Office, DASN/RDTE, Crosstalk, etc

* NSWCDD: Naval Surface Warfare Center, Dahlgren Division
Presentation Objectives and Content

❖ OBJECTIVES:
Provide awareness of:
− Common software system acquisition and engineering challenges
− Proven techniques for software system acquisition and engineering success
− Naval Software Improvement and Collaboration Efforts

❖ CONTENT
− Context information
− Challenges
− Keys to Success
− Improvement Interactions
− Summary
NSWCDD provides full spectrum system engineering & development for Naval warfare systems. This includes hands-on development of system and software requirements, architecture, design, and code; as well as system integration, test, and operational support responsibilities.
**Context: Naval Surface Warfare Center Dahlgren Division (NSWCDD)**

**Software Intensive Warfare System Development and Success**

<table>
<thead>
<tr>
<th>Rapid Development</th>
<th>Tactical Development</th>
<th>Strategic Development</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detect-Track-Engage Systems (Lasers, Guns, Non-Lethal)</td>
<td><strong>Surface Warfare Mission Module (Littoral Combat Ship)</strong>&lt;br&gt;Tomahawk Weapon Control System&lt;br&gt;Anti-Submarine Warfare System&lt;br&gt;Others..</td>
<td><strong>Submarine Ballistic Missile Mission Planning and Fire Control Systems</strong></td>
</tr>
</tbody>
</table>

*Also develop significant Simulations and Models: Ship Motion, Missile Flight, System Interfaces, Modeling/Sim Framework*

**Demonstrated Complex Mission-Critical Software Engineering Success**

- **Operational Success:** Thousands of successful Tomahawk Strikes and Battle Management System precision Strikes
- **Operational Success:** Detect-track-engage systems (Gunslinger, Wolfpack, Battle Management System,..)
- **Rapid technology transfer:** Deployed Laser Weapon System Quick Reaction Capability (LWC-QRC)
- Decades worth of successful ballistic missile test launches
- Award winning (Al Gore Hammer Award) modeling and simulations
- **High Quality Software:** Defect ratios consistently less than 1 defect per KSLOC (thousand lines of code)
- **Open Architected Systems:** Common, scalable, reliable, multi-platform capable software architectures

*Not the complete set of NSWCDD software systems and products; just a small sample*
Software Intensive Warfare Systems Challenges

**Software Size, Reliance, Complexity, and Cyber Threats**

**Cost, Schedule, & Technical Challenges**

**Government on-house applied Software Expertise**

Challenges include ever **increasing:**
- Demand for faster and cheaper development and delivery of systems to meet emergent needs & threats
- Pressure to “cut corners” and not utilize rigorous and disciplined data-driven best-practices/processes
- Cyber warfare threats and associated Information and Software Assurance requirements
- System and System-of-System (SOS) complexity and inter-dependencies
- Rapid evolution of software technologies and methodologies

**Primary Goal**

Consistently deliver high quality and reliable software systems as efficiently as possible

* Verified in documented performance result reports of DOD software system acquisition
Software Challenges: Project Execution

Limited use of mature data-driven best-practice based software project execution and continuous improvement

Poor software effort estimation and tracking processes (non metrics based)

Subjective assessment of cost, schedule, quality, and security versus metrics based

Program Leadership with limited applied sw expertise; treat software as a “black-box”

Poor communication

Poor requirements management; significant requirements volatility

Software architects not included in early system engineering phases

Poor software architecture: not enough time spent on the initial arch/design; jump-into-coding

Misuse of software prototyping (failure to throw away or formalize prototype code)

Lack of formalized risk management processes

Limited utilization of government in-house applied software expertise; over-reliance on industry

Not a complete set, but includes some of the common major contributors to cost, schedule and technical failures

Based on Numerous DOD/DON Studies/reports and first hand experience (300+ project reviews over last few years)
Software Challenges: Cyber Threats

Reactive cyber threat mitigation process
- Focus on Information Assurance (IA) versus **Software Assurance (Quality, Security, and Resiliency)**
- Numerous disjointed and non-coordinated “Cyber/IA/IT” organizations; proliferation of policy
- Numerous cyber-tools; challenges to acquire/fund/train/integrate tools into development processes
- Software security and resiliency were not designed-into legacy systems / cannot afford to re-architect
- RDTE systems treated same as Business IT systems; cumbersome certification req’s, limits flexibility and responsiveness

**Current “find-fix” cyber mitigation approach is re-active, slow, and costly**

Cyber vulnerabilities are increasing

Resiliency is not designed-in

Very expensive to fix vulnerabilities after Deployment

System Vulnerability Window: Months to Years

<table>
<thead>
<tr>
<th>Threat Created</th>
<th>Threat Deployed</th>
<th>Threat Disclosed</th>
<th>Threat Disclosed</th>
<th>Correction Available</th>
<th>Correction Deployed</th>
</tr>
</thead>
</table>

Unclassified  
Distribution A: Approved for public release; distribution is unlimited.
Keys to Software Engineering Success*

Mature Documented Data-Driven Processes

Work-Unit Based Software Effort Estimation and Tracking

Open Architecture

Agile/Rapid Development (Build-a-Little Test-a-Little)

Simulations and Data Extraction

Government and Industry Software-Development Teams

Engineer-In Software Assurance (Security, Quality and Resiliency)

Communication & Collaboration

* Not the complete set of best practices; just a selected subset of proven techniques
Keys to Software Success
Mature Documented andMeasured Processes

Capability Maturity Model Integrated (CMMI)
(A Framework for continuous improvement)

1. Define / Refine Process & Metrics
2. Estimate Cost, Schedule, Quality
3. Track Actual Cost, Sched, Quality
4. Analyze Metrics
5. Optimizing (Metrics Driven)

Technical Expertise coupled with Data Driven Processes facilitates consistent delivery of high quality software systems on schedule and within budget.
Keys to Project Execution Success
Best-Practice-Based System Engineering Processes

System Engineering includes Cost, Schedule, & Technical Performance

Best Processes:
- Project Planning
- Project Monitoring and Control
- Risk Management
- Integrated Product Teams (IPT)
- Configuration Management
- Requirements Development and Management
- Product Architecture
- Trade Studies
- Product Integration
- Verification
- Valiation

Of Projects deploying the most SE best practices, over 50% delivered higher project performance.

System Engineering best practices improves project execution.
System Engineering best practices include Project Planning & Control.
# Metrics Driven Project Execution

**GOAL:** Deliver High Quality Products on Schedule and Within Budget

<table>
<thead>
<tr>
<th>Question</th>
<th>Metric Examples (Not the complete set)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is the scope of the effort understood?</td>
<td>Requirements Impacted (New, Modified, Deleted)</td>
</tr>
<tr>
<td></td>
<td>Estimated Hours/Dollars per Activity (SE, SW, Test, etc)</td>
</tr>
<tr>
<td>Are the requirements understood, documented, allocated, and stable?</td>
<td>Requirements Volatility over time</td>
</tr>
<tr>
<td></td>
<td>Requirements Allocation to Org’s and Eng/Test Activities</td>
</tr>
<tr>
<td>Is the project adequately staffed?</td>
<td>Staffing profiles (by discipline and org)</td>
</tr>
<tr>
<td></td>
<td>Skill sets required versus on-board and available</td>
</tr>
<tr>
<td>Is the project on schedule?</td>
<td>Integrated Master Schedule and Detailed Schedules</td>
</tr>
<tr>
<td></td>
<td>Planned vs. Actual Progress w/ variance explanations</td>
</tr>
<tr>
<td>Is the project on budget?</td>
<td>Planned vs. Actual Cost w/ variance explanations</td>
</tr>
<tr>
<td>Is the project meeting quality goals?</td>
<td>Defect Detection and Closure Trends, Defect Ratios</td>
</tr>
<tr>
<td></td>
<td>Requirements and Tests Passed vs. failed</td>
</tr>
<tr>
<td>Is the project FORMALLY managing risks?</td>
<td>Open versus Closed Risk Trends</td>
</tr>
<tr>
<td>Is the project continuously improving?</td>
<td>Cost, Schedule, and Quality variance reductions</td>
</tr>
</tbody>
</table>

**Metrics should be:**

- Utilized in all activities, and by all organizations
- Proactively and regularly utilized (not just at milestone reviews)
- Documented, well defined, value added, and easily understood
- Supported by a Software Level Work Breakdown System (WBS)
- Capable of being rolled up to higher levels of abstraction
- Continuously assessed and improved
SLOC based estimation approach relies on several _unrealistic_ assumptions:
- Software engineers can accurately estimate hundred-of-thousands to millions of SLOC level efforts
- SLOC based productivity factors (SLOC per Hour) are based on accurate & relevant historical data
- SLOC productivity is indicative of other engineering activity productivity (Req's, Design, Test, etc)
- Constant effort relationship between SW activities and other engineering activities'
Software Success
Work-Unit Based Effort Estimation and Tracking

For Each Software Component and each SW development Activity (Requirements, Design, Code, Test):

**DEFINE WORK-UNITs (WU)**

Define Productivity (P) (Hours per Work Unit)

**ESTIMATE WORK UNITS**

- Utilize historical data from previous similar efforts

**CALCULATE EFFORT (Hours)**

Development Activity Hours = Estimated Work-Units * Productivity Factor

SW Component Hours = Sum of ALL Dev Activity Hours

Incremental Build Hours = Sum of ALL SW Component Hours

**IMPROVE ESTIMATION**

Compare Estimates to Actuals
- Revise Work Units
- Revise Productivity Factors

**TRACK PROGRESS**

Track Work Units Produced
Track Actual Productivity

Revise as required

**Work Unit Based Estimation & Tracking**

<table>
<thead>
<tr>
<th>Activity</th>
<th>Estimation Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQ's</td>
<td>Hours per Requirement, Hours per Interface, etc.</td>
</tr>
<tr>
<td>DESIGN</td>
<td>Hours per Object, Hours per DODAF view, etc.</td>
</tr>
<tr>
<td>CODE</td>
<td>Hours per Object, SLOC per Hour</td>
</tr>
<tr>
<td>TEST</td>
<td>Hours per Test Development, Hours per Execution</td>
</tr>
</tbody>
</table>

Define a “Work Unit(s)” per SE activity
Define associated productivity factor
Per each Work Unit

**System Eng “V” Chart**
Government Develops and owns the core Architecture, Interfaces, and software

Industry develops the “Plug and Play” Sensors, Launchers, Weapons Components

OA: A system composed primarily of common software that can be utilized across a wide range of platforms with minimal changes

Aligns with ASN RD&A Memo to the SYSCOMs/Chief of Naval Research “Use of In-House Engineering and Technical Requirements” 23 Feb 2012
Software Development Best Practices

ACTIVITIES:
- Mission Level Req’s
- Mission Level Architecture
- Mission Level Interfaces
- System Level Req’s
- System Level Architecture
- System Level Interfaces
- Early System Level Testing
  - Prototypes, Models, Simulations, Frameworks, ...
- Software Level Req’s
- Software Level Architecture
- Software Level Interfaces
- Early Software Component Level Testing
  - Representative (if possible) HW and SW Components;
  - Utilize Simulations and I/F drivers (GO AND FAULT PATHS)
  - Cyber security and System Resiliency Testing
- Software Components
- Software Components
- Early Software Component Level Testing
  - Representative (if possible) HW and SW Components;
  - Utilize Simulations and I/F drivers (GO AND FAULT PATHS)
  - Cyber security and System Resiliency Testing
- Integrated System Build
- Integrated System Test Builds
- Software / System Integration Testing
  - Actual and Representative HW and SW Components;
  - Augmented by Simulations and I/F drivers
  - Cyber Security, Penetration Testing, and System Resiliency Testing
- Platform based System Testing
  - Actual Hardware and Software
- Operational Support
- Well defined SW Support Activity
  - Quick recovery from exploited threats

Program Plan must include post-delivery support approach: Organizational responsibilities, funding, problem tracking

User-Centered and Risk Focused System/Software Engineering
- Complete Requirements Traceability and Configuration Management
- Formal Risk Management
- Adherence to Data-Driven Best Practices
- Early & Often Verification and Validation
  - Prototyping
  - Models and Simulations
  - Automated testing
  - Data Extraction & Analysis Tools

Design-In
- Software Security and Resiliency
- Automated Testing in early activities
- Utilize Cyber Security Tools

Defense-In-Depth:
- Protect-Detect-Isolate-Endure-Recover
  - Limit interfaces to external systems
  - Identify and harden (firewall) critical control points (interfaces)
  - Add processing to Detect cyber intrusions
  - Isolate and limit consequences (protect mission critical components)-
  - Endure through the Cyber intrusion to successfully complete the mission
  - Return the system to a trusted state

The Diagram represents a flowchart of the various stages and activities involved in software development, including mission level, system level, and software level requirements. It highlights the importance of early testing, including prototypes, models, and simulations, as well as the need for integrated system builds and testing. The diagram also emphasizes the importance of adhering to data-driven best practices and ensuring effective risk management throughout the development process. Additionally, it underscores the necessity of protective measures such as limiting interfaces to external systems, identifying and hardening critical control points, and adding processing to detect cyber intrusions. The final phase includes operational support and well-defined support activities to ensure a quick recovery from exploited threats.
**Rapid Software Development Goals**

- **Get Capability to Warfighter More Quickly**
  - Short-term releasibility
    - Providing an early version of the software with a subset of its ultimate functionality
  - Requirements flexibility
    - Ability to quickly change requirements while the software development is in progress.

**NOTE:** Agile development may not be the appropriate approach for all projects. Program Leaders must assess if Agile is the best approach for their specific program needs and structure.
Rapid Software Development

GOAL: Each sprint is working software, subset of capabilities, & capable of being delivered
Requires frequent regular communication between user, sponsor, & developers

- Rapid /Agile development is NOT an excuse to not have:
  - Documented data-driven processes
  - Cost, schedule, and technical performance measures
  - Requirement, Arch, Design and Test documentation

**EACH SPRINT**

- **Daily Scrum Meeting**
- **Product Backlog**
- **Sprint Backlog**
- **24 Hours**
- **2-4 Weeks**
- **Potentially Shippable Product Increment**
- **REQ**
- **Test**
- **Design**
- **Code**

Image available at www.mountaingoatsoftware.com/scrum

Copyright © 2005, Mountain Goat Software
**Software Success Simulations and Data Extraction**

**Key**
- Tactical Computer Software Configuration Items (CSCIs)
- Simulations
- Data Extraction

**Simulations**
Must support both “go path” and “fault’ path scenarios
- Fault Paths: Send data out of sequence, out of bounds, at high rates, etc

**Data Extraction**
All interface data, critical internal states and data

**Data Reduction Program**

- Must include effort to develop/modify simulations in cost and schedule planning
- Developed using disciplined processes; and ideally, by independent team
Many DOD/DON Program Managers have limited applied software expertise. Must have some gov’t engineers responsible for actual development (not just contractor oversight)
- Provides Program Manager with **business and technical advantages**
- Facilitates controlling cost: government is **not solely reliant on industry expertise**
- Provides Industry with a **true technical peer** to help negotiate cost, schedule and technical approach

**Government hands-on software development is required to:**
- Perform as a smart buyer and successfully team with industry
- Maintain expertise with the latest technologies

**Gov’t In-House Applied Expertise Pipeline**

**System Development Responsibility & Complexity**
- **Mission and Systems-Of-Systems Level**
- **System Level**
- **System Component Level**
- **System Sub-Components Level**

**Time / Experience**

**Hands-On Applied Expertise at all levels of complexity**

**Software Development Responsibility**
- **DEFINE System Req’s**
- **ARCHITECT System & Software**
- **DEVELOP System & Software**
- **INTEGRATE & TEST System**

**Gov’t Software Experts team with Industry SW Experts to Define, Design, Develop, and Deliver Software Systems**

**Teaming With Industry**

**Hands-On Development**
- Perform as a smart buyer and successfully team with industry
- Maintain expertise with the latest technologies
Government and Industry Software Development Teaming

Win - Win – Win - Win

- **Warfighter**
  - Faster receipt of capabilities
  - Increased capabilities
  - Higher quality and more reliable systems

- **Government Program Offices**
  - Improved Technology, Cost, and Schedule Estimates and Assessments
  - Increased and maintained corporate knowledge
  - Increased acquisition leverage and flexibility (more competition)

- **Industry**
  - Improved proposal assessments (smarter partner, not just lowest bid wins)
  - Reduced risk (smarter partner, improved requirements, government accountability)
  - *More profit* (less dollars on rework; increased system production and upgrades)

- **Taxpayer**
  - Better utilization of tax dollars
  - High quality, reliable, secure systems = Better protection of serving family members

*This software development teaming approach has been consistently successfully utilized for some of the Navy’s most critical warfare systems*
ACTIVITIES:

- Mission Level Req’s
- Mission Level Architecture
- Mission Level Interfaces
- System Level Req’s
- System Level Architecture
- System Level Interfaces
- Software Level Req’s
- Software Level Architecture
- Software Level Interfaces
- Software Components
- Software Components
- Integrated System
- Integrated System
- Integrated System
- System Testing
- Operational Support

Include senior level software architects during initial SE efforts

Throughout the system engineering process:
Utilize integrated multi-discipline product teams:
Include: User & Operator Rep’s, System Engineering, Software Engineering, Test, Logistics, Safety, etc.

Communicate, Communicate, Communicate!
Identify & mitigate cost, schedule and technical risks

Daily “stand-up” meetings

Weekly: discipline specific (SE, SW)
Focused cost, schedule and technical Performance and Risk meetings

Structured & Metrics Based Communication
Software Assurance: Best Practices
“Engineer-In” Quality, System Survivability, Security, and Resiliency

**GOAL:** Define, develop and deliver **high quality, secure and resilient systems**

Programs should designate a Chief SOFTWARE Architect in addition to Chief System Engineer

Define Assurance and Resiliency Requirements

**Architect/Design resilient “defense-in-depth” systems**

Protect-Detect-Isolate-Endure-Recover

Utilize tools to remove vulnerabilities during development

Define and utilize metrics and tools to quantify sw vulnerability, survivability, resiliency, and risk

Utilize data-driven best software engineering practices

Include independent Audits to ensure best practices

---

**Train the workforce on both sw best practices and Cyber vulnerabilities, threats, tools, secure coding, resilient design, implementation**
Software Assurance

Software reliance, complexity and cyber threats are increasing

Software Assurance focus must be more than just cyber security

Information Assurance (IA) compliance does NOT equal cyber security

There is no single “silver bullet” to ensure software quality and security

SW Assurance must be addressed throughout the system engineering life-cycle

Software Assurance requires application of software best practices, tools, and measures

Cannot “test-in” SW assurance, must “design-in” quality, security, and resiliency

Key to Success: Increased communication and collaboration

SW Assurance Definition (DOD):
The level of confidence that software functions as intended and is free of vulnerabilities, either intentionally or unintentionally designed or inserted as part of the software throughout the lifecycle. (per DoDI 5200.44)

SW Assurance Definition (Software Engineering Institute)
"Application of technologies and processes to achieve a required level of confidence that software systems and services function in the intended manner, are free from accidental or intentional vulnerabilities, provide security capabilities appropriate to the threat environment, and recover from intrusions and failures."

Software System Characteristics Goals (more than just “secure”)
- Secure, Safe, Reliable, Modular, Maintainable, Scalable, Portable, Defect Free
- Resilient- Meets war fighter mission critical needs despite cyber intrusion
Facilitate Awareness and Application of Best Software Engineering Practices via Increased Communication and Collaboration
Naval Software Community of Practice (SW COP) Executive Summary

- **Background:** In 2009, NSWCDD initiated the establishment of a Naval Software Community of Practice (SW COP)

- **Goal:** Improve Naval warfare software system’s cost, schedule, and technical performance.
  - Share best practices, processes, tools, techniques and artifacts
  - Provide resources to help solve complex technical software problems
  - Provide access to software engineers with specialized expertise
  - Provide awareness of software laws, policies, guides and requirements
  - Promote maintaining government in-house applied software expertise

- **Approach:** Increase communication and collaboration between government sw experts

- **Participation** 280+ participating software experts from 17 different organizations

- **Results:** 800+ artifacts posted to the knowledge sharing site

- **Results:** 850+ hours saved via collaboration

**PRODUCTS:** SW COP provided inputs to DoD and DoN policies and guides:

- OSD Program Protection Policy (PPP) SW Assurance (IN-WORK)
- DOD SEI SW Sustainment Study (In-Work)
- Office of the Secretary of Defense (OSD) Defense Acquisition Guide (DAG)
- Naval Open Architecture (OA) Contracts Guide
- Naval Open Architecture (OA) Metrics Guide
- Naval System Engineering Guide: Software Sections
- Naval Guidebook for Acquisition of Software Intensive Systems (SW Guidebook)
Keys to Software System Acquisition Success

Summary

Gov’t SW engineers
Hands-On Full Spectrum Engineering

Technical Expertise coupled with Data-Driven Continuous Improvement

Applied Technical Expertise

1. Define & Refine Process & Metrics
2. Estimate: Cost, Schedule, Quality
3. Track Actuals: Cost, Schedule, Quality
4. Analyze Metrics

Management Processes

METRICS DRIVEN

High quality products Delivered on time & within budget
Continuous data-driven improvement

Technical Processes

Project Reviews
Quarterly Execution Reviews

I. SYSTEM ANALYSIS & CONTROL

1. Define & Refine Process & Metrics
2. Estimate: Cost, Schedule, Quality
3. Track Actuals: Cost, Schedule, Quality
4. Analyze Metrics

Applied Technical Expertise

Gov’t Experts Teaming With Industry

Software Development Responsibility

HANDS-ON DEVELOPMENT

DEFINE System Req’s

ARCHITECT System & Software

DEVELOP System & Software

INT/TEST System

RESULT:
Government Owned & Controlled System Arch & Software

Best Practices

Maintaining and utilizing gov’t in-house applied software engineering expertise
Disciplined data-driven project execution and continuous improvement
Disciplined requirements management
Formal Risk Management
Open architected and defense-in-depth architected systems
Agile / Rapid / Build-a-little Test-a-little development methodologies
Design-in Software quality, Security and Resiliency
Increased communication and collaboration; sharing of best-practices

Utilize Data-Driven Best Practices and Maintain In-House Applied Software Expertise
Data Driven Success

“In God we trust, all others must bring data”.
W.E. Demming

“For every opinion, there is an equal and opposite opinion; but, for every fact there is not an equal and opposite fact”
L. Albuquerque

“Without data, you are just another person with an opinion”.
Unknown

“You cannot expect what you do not inspect”
MARCOR proverb

“Trust but verify”
Ronald Reagan
Software Challenges
Increasing Software Size and Complexity

Many Warfare System Program Managers do not have applied expertise with software engineering. Software is often treated as a “black box”; software size and complexity is not understood.
Software Challenges
Dept Of Defense (DOD) Software System Acquisition Results

DOD SOFTWARE SYSTEM ACQUISITION PERFORMANCE

- 100% Capabilities Delivered
- 96% Development $ Spent
- 50% Operational Testing
- 50% Exceeded Nun-McCurdy
- 84% On Schedule And Budget

- 39% Not Delivered
- 40% Spent on Rework Due to SW Problems
- 50% Failed Initial Op-Tests
- 50% Exceeded Cost Threshold

Failures: Cost, Schedule or Technical Performance
Success: Cost, Schedule or Technical Performance

References:
Secretary of Defense (SECDEF), SECDEF Memo: "Department of the Navy Acquisition", December 2008.
Senator Carl Levin, U.S. Senate Committee of Armed Services Press Release, March 2009

Software intensive warfare system engineering efforts are not consistently successful with regards to cost, schedule, and technical performance
**Assertion:**
Cannot assume that all cyber vulnerabilities and threat vectors are known.
Cannot assume that systems are 100% secure and can be protected from penetration.

**Utilize Defense-In-Depth and Design-In System Resiliency**

**Protect-Detect-Isolate-Endure-Recover**
- Limit interfaces to external systems
- Identify and harden (firewall) critical control points (interfaces)
- Add processing to Detect cyber intrusions
- Isolate and limit consequences (protect mission critical components)
- Endure through the Cyber intrusion to successfully complete the mission
- Return the system to a trusted state
Software Success
Open Architecture (OA) Characteristics

Open Architected Software System:
A system composed primarily of common software that can be utilized across a wide range of platforms with minimal changes

* Reference: OA Architectural Principles and Guidelines v 1.5.6, 2008, IBM, Eric M. Nelson, Acquisition Community Website (AC) DAU Navy OA Website

Composability
The System Provides Recombinant Components that can be Selected and Assembled in Various Combinations to Satisfy Specific Requirements

Interoperability
Ability of Two or More Subsystem to Exchange Information and Utilize that Information

Open Standards
Standards that are Widely Used, Consensus Based, Published and Maintained by Recognized Industry Standards Organizations

Maintainability
The Ease With Which Maintenance of a Functional Unit can be Performed in Accordance With Prescribed Requirements

Extensibility
Ability to add new Capabilities to System Components, or to add Components and Subsystems to a System

Modularity
Partitioning into Discrete, Scalable, and Self-Contained Units of Functionality, With Well Defined Interfaces

Diagram Key
- is Enabled by
- is Facilitated by

Open Systems Facilitate Achieving the Following:
- Reduce life cycle costs, Reduce acquisition time
- Increase system reliability, maintainability, and quality
- Increase competition (small business develops components)
Best Practices: Communication

Well defined and *documented* Statement of Work (SOW):
- Clearly Defined Tasking, Roles, Responsibilities and *Deliverables*
- Protect and Ensure government ownership rights of software deliverables

Establish and maintain frequent periodic structured communication
- Structured (standardized) agenda and data/metric based reporting (not subjective)
- Maintain planned vs. actual cost, schedule, and technical performance metrics

Document and track Risks: Utilize Risk Management Tools (e.g. Risk Exchange)

Ensure Cross Organizational and Engineering Discipline Communication
- Utilize Integrated Product Teams (IPTs): Customer, Users, System Eng’s, Software, Test, etc

Daily “stand-up” meetings with Program and Technical Leads: Short, Concise, Risk focused

Weekly specific engineering discipline (e.g. SE, SW) meetings
- Cost, Schedule, Technical Performance and Risk meetings

Event Driven Project Reviews: Technical Reviews, Delivery Readiness, Project Completion

*Communicate, Communicate, Communicate!*
*Identify, communicate, and mitigate cost, schedule, and technical risks*