National Aeronautics and Space Administration

DRAFT: Release version 0.1C | March 5, 2011

NASA Software Engineering Handbook

Supplement to the NASA Procedural Requirement (NPR) 7150.2A

Table of Contents

1. Introduction

We are glad you have come to the NASA Software Engineering Handbook site. The purpose of this site is to provide key insights to you, a Software Engineering professional. We plan on two phases of release: the first with 30% material in February 2011, and with 80% material in October 2011.

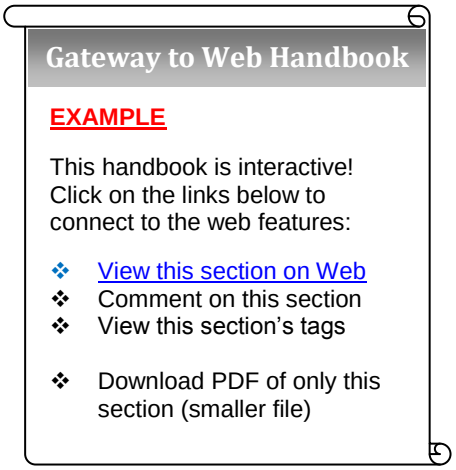
To view a presentation that was given on the handbook development to the Software Engineering Working Group in August 2010, click here.

How is this different than any other NASA handbook?

The Software Engineering handbook will have two components. The first is a PDF/printable version for those who wish to use the material in a more traditional way, which you are reading right now. We are also developing this web version as an interactive and dynamic version of the same material. We plan on utilizing web technologies, such as tagging (folksonomies), social commenting, and web editability and versioning to enhance the experience of what a paperback handbook provides.

The web version is available at http://nasa7150.onconfluence.com and will be moved to a NASA.gov domain by the Fall 2011 Software Working Group Face to Face.

On the web, we are already accepting comments at the very bottom of this page. You may leave an anonymous comment, but please use responsibly and according to the Code of Conduct.



Gateway to Web Handbook

Each section of this draft handbook has a ***Gateway to Web Handbook*** bubble in the right hand corner of the first page. Eventually, every bulletted item in the bubble will be a link as the each name indicates. But for now only the first link works, which is the ***View this section on Web***. If you click on this link you can view the material on the NASA Software Engineering Handbook Website (powered by the 7150 Wiki).

Special Topic Material

There are a total of 37 special topics which will be chapters within the handbook. A partial list is as follows:

7.1 - 7150.2A Definitions & References, 7.2 - Classification Tool and Safety Critical Assessment Tool, 7.3 - Lifecycle Management, 7.4 - Entrance / Exit Criteria for 7150.2A, 7.5 - Documentation Products Maturity (List of material maturity by phase of mission), 7.6-8 - 7150.2A's Traceability to Other NPRs, 7.9 - Software Acquisition, 7.11 - Use of COTS, GOTS, MOTS, 7.12 - Flow down of NPR requirements on contracts and to other centers in multi center projects, 7.16 - Transitioning to a higher class, 7.17 - Explanation of enforcement of NPR requirements, 7.18 - Compliance matrices

Material by SWE Requirement Number

The NASA Software Engineering standards are laid out in the NPR 7150.2A document (click here to view it on the web). Within the document, there are requirements numbering up to 130. We will be producing material for each of these requirements in the following areas: Guidance, Rationale, Tools Available, Links, and Guidance for Small Project.

What's in Release 0.1C?

Correlating with John Kelly's email the week before the Software Working Group Face to Face (March 1-3, 2011), these are the sections of the handbook which are ready for review and have been incorporated into this document.

Title from John Kelly Email	Correlating Chapter (in this Document)
1. Session 1 E&E SSC F2F 3-1-2011 V1.0.ppt	Chapter 3
2. Use of Commercial, Government, Legacy.pptx	Chapter 4
3. F2F_Criteria_Pitch_DJG_2011Feb18.ppt	Chapter 3
4. SWG F2F Presentation on Acquisition_w notes_20110221.ppt	Chapter 5
5. SWG F2F Presentation on Entry-Exit_w notes_20110224.ppt	Chapter 3
6. SWG F2F Presentation on Transition_w notes_20110128.ppt	Chapter 6
7. SWE-029 Validation Planning	Chapter 7
8. SWE-033 Acquisition Assessment	Chapter 8

Contributors

Many have contributed to this early draft of the Software Engineering Handbook. They are as follows:

John Kelly
Kevin Carmichael
Dave York
Kathy Malnick
Tommy Tayman
Lee Jackson
Jon Verville
Dan Gauntner

Also, special thanks to the members of the NASA Software Working Group (NSWG) for input, review, and contributions.

2. Lifecycle Review Entry/Exit Criteria Guidance

This section identifies criteria for the entrance into and the successful completion of one of the 13 lifecycle reviews from NPR 7123.1 Appendix G. It is organized by the 13 reviews chosen and includes the following information: entrance criteria, exit criteria, software community responsibilities (e.g., SDP/SMP), software community contributions to system activities/products (e.g., Project Plan).

Material was also added and adapted from these other sources:

- NPR 7120.5D (NM 7120-81)
- Center documents & PALs: ARC, JPL, GSFC, MSFC, SSC
- Defense Acquisition Guidebook

Gateway to Web Handbook

Prototype: only the top link works (view section on web)

This handbook is interactive! Click on the links below to connect to the web features:

- View this section on Web
- Comment on this section
- View this section's tags
- Download PDF of only this section (smaller file)

Notes: The software requirements review (SwRR) is not included in 7123.1, but is often used in software projects, so it is included here. The following reviews were not included because they did not have any apparent correlation to 7150.2A: Program/System Requirements Review, Program/System Definition Review, Post-Launch Assessment Review, Critical Event Readiness Review, Post-Flight Assessment Review, Decommission Review, Periodic Technical Review. (Tables G-1,2,15-19)

Mission Concept Review (MCR)

Entrance Criteria

- Need for mission clearly identified
- Mission goals and objectives clearly defined and stated; unambiguous and internally consistent
- Analysis of alternative concepts (showing at least one feasible)
- Concept of operations
- Preliminary risk assessment, including technologies and associated risk management/mitigation strategies and options
- Conceptual test and evaluation strategy
- Preliminary technical plans to achieve next phase
- Conceptual life-cycle
- Preliminary Software Management Plan (SMP)
- The preliminary set of requirements to meet the mission objectives
- The mission is feasible
  - A solution has been identified that is technically feasible
  - A rough cost estimate is within an acceptable cost range
- The cost and schedule estimates are credible
- An updated technical search was done to identify existing assets or products that could satisfy the mission or parts of the mission

Exit/Success Criteria

- Technical planning is sufficient to proceed to the next phase
- Risk and mitigation strategies have been identified and are acceptable based on technical risk assessments
- As applicable,
  - Science objectives are clearly understood and comprehensively defined
  - Preliminary mission requirements are traceable to science objectives

    - Operations concept clearly supports achievement of science objectives
- Conceptual system design meets mission requirements, and the various system elements are compatible
- Technology dependencies are understood, and alternative strategies for achievement of requirements are understood
- Conceptual system design meets mission requirements, and the various system elements are compatible
- Technology dependencies are understood, and alternative strategies for achievement of requirements are understood

Systems Requirements Review (SRR)

Entrance Criteria

- Successful completion of the MCR and responses made to all MCR Requests for Actions (RFAs) and Review Item Discrepancies (RIDs)
- A preliminary SRR agenda, success criteria, and charge to the board have been agreed to by the technical team, project manager, and review chair
- Technical products made available to participants prior to SRR (noted in this list)
- System requirements document
- Preliminary system requirements allocation to next lower level system
- System software functionality description
- Updated concept of operations
- Updated mission requirements, if applicable
- Software inputs / contributions to
    - Baselined Systems Engineering Management Plan (SEMP)
    - Preliminary Project Plan
    - System safety and mission assurance plan
    - Risk management plan
        - Updated risk assessment and mitigations (including Probabilistic Risk Assessment (PRA) as applicable)
    - Technology Development Maturity Assessment Plan
    - Logistics documentation (e.g., preliminary maintenance plan)
    - Preliminary human rating plan, if applicable
    - Initial document tree
- Lessons Learned
    - Review of existing Lessons Learned from previous projects
    - Lessons Learned captured from software areas of the project; indicate the problem or success that generated the Lesson Learned, what the Lesson Learned was, and its applicability to future projects
    - Confirmation that Lessons Learned added to Lessons Learned database

Exit/Success Criteria

- Process for allocation and control of requirements throughout all levels deemed sound; plan defined to complete the definition activity within schedule constraints
- Requirements definition is complete with respect to top-level mission and science requirements; interfaces with external entities and between major internal elements have been defined
- Requirements allocation and flow down of key driving requirements defined down to subsystems
- Preliminary allocation of system requirements to hardware, human, and software subsystems
- Preliminary approaches determined for how requirements will be verified and validated down to the subsystem level

- Major risks identified and technically assessed, and viable mitigation strategies defined
- Requirements and selected concept will satisfy the mission
- System requirements, approved material solution, available product/process technology, and program resources form a satisfactory basis for proceeding into the development phase

Software Requirements Review (SwRR)

Entrance Criteria

- Successful completion of the SRR and responses made to all SRR Requests for Actions (RFAs) and Review Item Discrepancies (RIDs)
- A preliminary SwRR agenda, success criteria, and charge to the board have been agreed to by the technical team, project manager, and review chair
- Technical products made available to participants prior to SwRR (noted in this list)
- Updated concept of operations
- Preliminary system requirements allocation to software
- Software requirements (SRS)
    - Complete, consistent, feasible, testable, and traceable
    - Identify test, delivery and quality requirements, and are understandable
- Functional requirements
    - High-level requirements for each functional area
    - Block diagram of the major software components in each functional area, their interfaces and data flows
    - Definition of relevant operational modes (e.g., nominal, critical, contingency)
    - Critical and/or controversial requirements, including safety-critical requirements, open issues, and areas of concern
    - Requirements needing clarification or additional information
- Traceability Matrix (bidirectional)
    - Requirements to higher-level requirements
    - Requirements to build and system level tests
- Performance requirements
    - Performance requirements for the software
    - Critical timing relationships and constraints
- Software Interface Specifications (SISs - requirements portion)
- Software requirements and interface requirements have been analyzed and specified
- Computer resource estimates and margins (memory, bus, throughput)
- Software Quality Assurance Plan (SQAP)
- Software QA organization structured with independent reporting relationship outside development group
- Updated PHA/Software Assurance Classification Report (SACR), Software Safety Litmus Test
- Review for technical and economic feasibility completed for allocation of functions at the (sub)system level to hardware, firmware, and software
- Design Constraints
- Design strategy
    - Explanation of design drivers and design decisions that have been made, including software architecture, operating systems, reuse of existing software, and selection of COTS components
    - Resource goals and preliminary sizing estimates (incl. timing and database storage) in the context of available hardware allocations; strategies for measuring and tracking resource utilization
    - Initial Build Plan

- Risk management plan
  - Risks that may impact cost, schedule and technical goals completed
- Configuration Management Plan addressing:
  - Configuration identification, change control, status accounting, and configuration audits
- SDP / SMP updated for corresponding architectural design and test development activities
  - Personnel identified (quantity, names, assignment duration, required skills)
  - Organizational responsibilities and interfaces
  - All computer programs identified, their development schedules compatible, their dependencies evident in schedules, and supporting resource allocations made
  - Updated cost estimate
  - Milestones are verifiable and achievable
  - Schedules for development of all computer programs, and procedures for monitoring and reporting their status
  - Processes and metrics for program success
  - Management methods and controls for design & development
  - Programming languages, security requirements, operational and support concepts identified
  - Preliminary high level software architecture
- Qualification requirements
  - Overall software test strategy, including the test levels (unit, integration, build, and system-level testing), test types (interface, load/stress, regression), and test tools
  - Software development and test environments, including processors, operating systems, communications equipment, simulators and their fidelity
  - Test facilities, needs and capabilities
  - Methodology for verifying the system requirements and acceptance criteria
  - Test tool requirements and development plans
- Preliminary software V&V plan
- Peer reviews completed: SMP, s/w requirements, V&V plans, preliminary s/w system architectural design (if identified for peer review/inspection in s/w development plans)
- Make-buy decisions supported by analysis
- Analyses completed, as applicable:
  - Functional analyses
  - Testability
  - Operability
  - Failure modes and effects analyses
  - Reliability engineering
  - Systems safety and hazards
  - Life-cycle costs
  - Security
- Trade-off and design decisions completed and reviewed, as applicable, for:
  - Inherited capabilities
  - New technologies
  - Programming language selection
  - Sizing and timing budget
  - Design methods and tool selection
  - Programming standards and conventions
  - Database conceptual design
- Software User's Guide/Operator's Manual (UG/SOM - user's guide portion)

Exit/Success Criteria

- Software requirements determined to be clear, complete, consistent, feasible, traceable, testable

- SMP, software requirements, interface requirements, V&V plans are adequate and feasible basis for architectural design activities and are approved, baselined and placed under configuration management
- Requirements and performance requirements defined, testable, and consistent with cost, schedule, risk, technology readiness, and other constraints
- System requirements, approved material solution, available product/process technology, and program resources form a satisfactory basis for proceeding into the development phase
- All SwRR RIDs and actions are documented with resolution plans and authorization received to proceed to software architecture design

Mission Definition Review (MDR)

Entrance Criteria

- Successful completion of the previous review (typically SRR) and responses made to all Requests for Actions (RFAs) and Review Item Discrepancies (RIDs)
- Preliminary agenda, success criteria, and charge to the board have been agreed to by the technical team, project manager, and review chair
- Technical products made available to participants prior to SDR (noted in this list)
- System architecture, including software
- Preferred software solution definition including major tradeoffs and options
- Updated baselined documentation, as required
- Preliminary functional baseline (with supporting trade-off analyses and data)
- Preliminary system software functional requirements
- Updated risk management plan (could be part of SDP/SMP)
    - Updated software risk assessment and mitigations (including Probabilistic Risk Assessment (PRA), as applicable)
- Updated SDP/SMP
    - Updated technology development, maturity, and assessment plan
    - Updated cost and schedule data
    - Work Breakdown Structure
- Updated logistics documentation
- Software verification and validation plan
- Software requirements document(s)
- Interface requirements documents (including software)
- Technical resource utilization estimates and margins
- Updated preliminary software safety analysis
- Project Software Data Dictionary
- Project Software Configuration Management Plan
- Project Software Assurance Plan
- Project Software Maintenance Plan
- Software inputs / contributions to
    - Systems Engineering Management Plan (SEMP) changes, if any
    - Based on system complexity, updated human rating plan
    - Flow down of system requirements to all software functional elements of the system

Exit/Success Criteria

- Software requirements, including mission success criteria and any sponsor-imposed constraints, are defined and form the basis for the proposed conceptual design
- All software technical requirements are allocated and the flow down to subsystems is adequate; requirements, design approaches, and conceptual design will fulfill the mission needs

consistent with the available resources (cost, schedule, throughput, and sizing)
- Requirements process is sound and can reasonably be expected to continue to identify and flow detailed requirements in a manner timely for development
- Technical approach is credible and responsive to the identified requirements
- Technical plans have been updated, as necessary
- Tradeoffs are completed, and those planned for Phase B adequately address the option space
- Significant development, mission, and safety risks are identified and technically assessed, and a process and resources exist to manage the risks
- Adequate planning exists for the development of any enabling new technology
- Operations concept is consistent with proposed design concept(s) and in alignment with the mission requirements
- All allocated requirements are verifiable and traceable to their corresponding system level requirement
- Preliminary verification approaches are agreed upon
- Requisite level of detail and resources are available to support the acquisition and development plan within existing constraints
- A software system is defined which satisfies all of the system requirements assigned to software
- All of these software system requirements are traceable to either mission objectives, concept of operations, or interface requirements
- Monitoring processes/practices are in place to create software system within planned technical, schedule, cost, effort, and quality capabilities

SYSTEM DEFINITION REVIEW (SDR)

Entrance Criteria

- Successful completion of the previous review (typically SRR) and responses made to all Requests for Actions (RFAs) and Review Item Discrepancies (RIDs)
- Preliminary agenda, success criteria, and charge to the board have been agreed to by the technical team, project manager, and review chair
- Technical products made available to participants prior to SDR (noted in this list)
- System architecture, including software
- Preferred software solution definition including major tradeoffs and options
- Updated baselined documentation, as required
- Preliminary functional baseline (with supporting trade-off analyses and data)
- Preliminary system software functional requirements
- Updated risk management plan (could be part of SDP/SMP)
    - Updated software risk assessment and mitigations (including Probabilistic Risk Assessment (PRA), as applicable)
- Updated SDP/SMP
    - Updated technology development, maturity, and assessment plan
    - Updated cost and schedule data
    - Work Breakdown Structure
- Updated logistics documentation
- Software verification and validation plan
- Software requirements document(s)
- Interface requirements documents (including software)
- Technical resource utilization estimates and margins
- Updated preliminary software safety analysis
- Project Software Data Dictionary
- Project Software Configuration Management Plan
- Project Software Assurance Plan
- Project Software Maintenance Plan
- Software inputs / contributions to
    - Systems Engineering Management Plan (SEMP) changes, if any
    - Based on system complexity, updated human rating plan
    - Flow down of system requirements to all software functional elements of the system
- Software requirements, including mission success criteria and any sponsor-imposed constraints, are defined and form the basis for the proposed conceptual design
- All software technical requirements are allocated and the flow down to subsystems is adequate; requirements, design approaches, and conceptual design will fulfill the mission needs consistent with the available resources (cost, schedule, throughput, and sizing)
- Requirements process is sound and can reasonably be expected to continue to identify and flow detailed requirements in a manner timely for development
- Technical approach is credible and responsive to the identified requirements

Exit/Success Criteria

- Software requirements, including mission success criteria and any sponsor-imposed constraints, are defined and form the basis for the proposed conceptual design
- All software technical requirements are allocated and the flow down to subsystems is adequate; requirements, design approaches, and conceptual design will fulfill the mission needs consistent with the available resources (cost, schedule, throughput, and sizing)
- Requirements process is sound and can reasonably be expected to continue to identify and flow detailed requirements in a manner timely for development
- Technical approach is credible and responsive to the identified requirements
- Technical plans have been updated, as necessary
- Tradeoffs are completed, and those planned for Phase B adequately address the option space
- Significant development, mission, and safety risks are identified and technically assessed, and a process and resources exist to manage the risks
- Adequate planning exists for the development of any enabling new technology
- Operations concept is consistent with proposed design concept(s) and in alignment with the mission requirements
- All allocated requirements are verifiable and traceable to their corresponding system level requirement
- Preliminary verification approaches are agreed upon
- Requisite level of detail and resources are available to support the acquisition and development plan within existing constraints
- A software system is defined which satisfies all of the system requirements assigned to software
- All of these software system requirements are traceable to either mission objectives, concept of operations, or interface requirements
- Monitoring processes/practices are in place to create software system within planned technical, schedule, cost, effort, and quality capabilities

Preliminary Design Review (PDR)

Entrance Criteria

- Successful completion of the SDR or MDR and responses made to all SDR or MDR Requests for Actions (RFAs) and Review Item Discrepancies (RIDs), or a timely closure plan exists for those remaining open
- Preliminary agenda, success criteria, and charge to the board have been agreed to by the technical team, project manager, and review chair
- Technical products made available to participants prior to PDR (noted in this list)
- Updated baselined documentation, as required
- Updated technology development maturity assessment plan
- Updated risk assessment and mitigation
- Updated cost and schedule data
- Updated logistics documentation, as required
- Applicable technical plans (e.g., technical performance measurement plan, payload-to-carrier integration plan, producibility / manufacturability program plan, reliability program plan, quality assurance plan)
- Applicable standards
- Interface control documents
- Software V&V Plan
- Technical resource utilization estimates and margins
    - Storage or memory resource allocations developed allocating those resources to each software segment in the architecture

- Updated SDP/SMP
  - Work Breakdown Structure
- Preliminary Traceability Matrix to CSCI level, including V&V trace
  - safety-critical requirements highlighted
  - Requirements allocated to components of the architecture (to CSCI level)
- SDD and Traceability Matrix review by test team completed and SDD updated as needed
- SMP updated for the corresponding detailed design activities
- Software inputs or contributions to the updated Project Plan
- Supplier documentation
  - Software Data Dictionary(s)
  - Software Classification(s)
  - Software Development or Management Plan(s) [with V&V separate]
  - Software Configuration Management Plan(s)
  - Software Assurance Plan(s)
  - Software Maintenance Plan(s)
- Revised SRS
  - Software requirements to CSCI level
  - Subsystem and lower-level technical requirements
  - Requirements for reuse of existing software, reuse analysis
  - Performance requirements, including memory, bus, CPU requirements
  - Quality requirements, e.g., reliability, usability, or maintainability requirements
  - Safety requirements
  - Security requirements
  - Derived requirements
- Revised Operational Concepts, as applicable
  - Normal operations scenarios
  - Fault detection, isolation and recovery (FDIR) strategy
  - Hazard reduction strategies
- Lessons Learned
  - Review of existing Lessons Learned from previous projects
  - Lessons Learned captured from software areas of the project; indicate the problem or success that generated the Lesson Learned, what the Lesson Learned was, and its applicability to future projects
  - Confirmation that Lessons Learned added to Lessons Learned database
- Trade studies
  - Addressing COTS, reuse, etc.
  - Trade-off analysis and data supporting design, as required
  - Documented Alternative Design Solutions and Selection Criteria
- Documented Solutions, Analysis, Decision, and Rationale
  - Inherited capabilities identified and compatible with the designs
- Preliminary Software Design Document (SDD)
- Subsystem design specifications for each configuration item (h/w and s/w)
- Completed definition of the software architecture and preliminary database design description, as applicable
  - External interfaces and end-to-end data flow
  - Design drivers (e.g., performance, reliability, usability, hardware considerations)
  - Overview of software architecture, including context diagram
  - List of subsystems, tasks, or major components – e.g., user interface, database, task management
  - Functional allocations, descriptions of major modules, and internal interfaces

    - Safety considerations in the design elements and interfaces
    - Design verification approach, e.g., prototyping, inspection, peer review
    - Architectural design verified via operational scenarios to include required functionality, operating modes, and states
- Safety analyses and plans
    - Matrix showing each subsystem/task/component's software classification (per NPR 7150.2A), its safety classification (per NASA-STD-8719.13B), the rationale for the classifications, and the status of the classifications' approval by Software Assurance and management
    - Updated PHA, Software Safety Litmus Test, if necessary
    - Approved SMP/ PHA/Software Assurance Classification Report (SACR)
- Analyses completed:
    - Partitioning analysis (modularity)
    - Executive control and Start/Recovery
    - Control and Data flow analysis
    - Operability
    - Failure modes and effects analyses
- Results of prototyping factored into architectural design
- Prototype software, if necessary
- Critical components identified and trial coding scheduled
- Human engineering aspects of design addressed with solutions acceptable to potential users
- Developmental tools and facility requirements identified and plans made and actions taken to ensure their availability when needed
- Test tools and facility requirements identified with plans and actions to ensure their availability when needed
- Test group involved in requirements and design analysis
- Security and supportability requirements factored into the design
- Metrics established and gathered to measure software development progress
- Procedures and tools developed for mechanizing management and configuration management plans
- Configuration Control Board established for software (and change control procedures working)
- Configuration management system understood by those who must use it
- Library established for storing, controlling and distributing software products; library procedures understood and working
- Independent software quality assurance group formed and contributing as a team member to the design and test activities
- Interdisciplinary teams working design issues that cross (sub)system component boundaries (software, hardware, etc.)
- Peer reviews completed: SRS , software architectural design (if identified for s/w peer review/inspection in s/w development plans), integration test plans
- Status of change requests

Exit/Success Criteria

- Top-level requirements including mission success criteria, Technical Performance Measures (TPMs), and any sponsor-imposed constraints are agreed upon, finalized, stated clearly, and consistent with preliminary design
- Flow down of verifiable requirements is complete and proper or, if not, an adequate plan exists for timely resolution of open items; requirements are traceable to mission goals and objectives
    - All supplier software requirements are verifiable

- ⦿ Preliminary design is expected to meet the functional and performance requirements at an acceptable level of risk
- ⦿ Definition of technical interfaces is consistent with overall technical maturity and provides an acceptable level of risk
- ⦿ Adequate technical interfaces are consistent with the overall technical maturity and provide an acceptable level of risk
- ⦿ Adequate technical margins exist with respect to TPMs
- ⦿ Any required new technology has been developed to an adequate state of readiness, or back-up options exist and are supported to make them a viable alternative
- ⦿ Project risks are understood and credibly assessed; plans, process, and resources exist to effectively manage them
- ⦿ Operational concept is technically sound, includes (where appropriate) human factors, and includes flow down of requirements for its execution
- ⦿ All RIDs/actions are completed and customer approval to proceed to detailed design phase
- ⦿ Proposed design approach has sufficient maturity to proceed to final design
    - Subsystem requirements, subsystem preliminary design, results of peer reviews, and plans for development, testing and evaluation form a satisfactory basis for proceeding into detailed design and test procedure development
- ⦿ SMP, the software architectural design, and integration test plans adequate and feasible to support software detailed design
- ⦿ Products (listed above) are approved, baselined and placed under configuration management
- ⦿ Software inputs / contributions to
    - Safety and mission assurance (e.g., safety, reliability, maintainability, quality, and EEE parts) adequately addressed in preliminary designs and any applicable S&MA products (e.g., Probabilistic Risk Assessment (PRA), system safety analysis, and failure modes and effects analysis) have been approved
- Management processes used by the mission team are sufficient to develop and operate the mission
- Cost estimates and schedules indicate that the mission will be ready to launch and operate on time and within budget, and that the control processes are adequate to ensure remaining within allocated resources

CRITICAL DESIGN REVIEW (CDR)

Entrance Criteria

- Successful completion of the previous review (typically PDR) and responses made to all Requests for Actions (RFAs) and Review Item Discrepancies (RIDs), or a timely closure plan exists for those remaining open
- Preliminary agenda, success criteria, and charge to the board have been agreed to by technical team, project manager, and review chair
- Technical products made available to participants prior to CDR (noted in this list)
- Updated baselined documents, as required
- Technical data package (e.g., integrated schematics, Spares provisioning list, interface control documents, engineering analyses, and specifications)
- Updated Technology Development Maturity Assessment Plan
- SMP updated for implementation and unit test activities
    - Updated Work Breakdown Structure
    - Updated cost and schedule data
- Progress against software management plans
- Plan for milestone and peer reviews, walkthroughs, and external reviews
- Documentation plan, including each document's status and when it will be baselined
- Software requirements, management process, including documents used and produced, and V&V plan are baselined
- Preliminary NPR 7150.2 compliance matrix
- Design process, including methodology and standards used, design documentation produced, inspections and reviews
- Implementation process, incl. standards, review process, problem reporting, unit test, integration
- Management procedures and tools for measuring and reporting progress available and working
- Software measurements on planned and actual regarding product size, cost, schedule, effort, and defect
- Procedures established and working for software quality assurance and quality an integral part of the product being produced
- Updated logistics documentation
- Staffing-up problems being addressed and contingency plans in place
- IT Security Requirements (Mission-specific)
- Software design document(s) (including interface design documents, detailed design and unit test)
- Command and telemetry list
- Final Design Solution, Evaluation, and Rationale
    - Documented Make, Buy, and/or Reuse, Analysis, Criteria, and Rationale
    - Reused/heritage software or functionality from previous projects; necessary modifications
- Final Architecture Definition
- System design diagram (e.g., Level 0 data flow diagram or UML)
    - For each task in the system design diagram
    - Design diagrams for the task
    - Description of functionality and operational modes
    - Safety considerations addressed in the design
    - Resource and utilization constraints (e.g., CPU, memory); how the software will adapt to changing margin constraints; performance estimates
    - Data storage concepts and structures
- Data flow diagrams

- Identification and formats of input and output data
- Interrupts and/or exception handling, including event, FDC, and error messages
- IT Security features (design features)
- Detailed description of software operation and flow
- Operational limits and constraints
- Technical resource utilization estimates and margins
    - Detailed timing and storage allocation compiled
- Analyses completed:
    - Algorithm accuracy
    - Critical timing and sequence control
    - Dimensional analysis (such as consistency of array dimensions)
    - Singularity Analysis (such as division by zero)
    - Undesired event handling
    - Operability
    - Failure modes and effects analyses
- Final status and results of analyses
- Algorithms sufficient to satisfy their requirements
- Failure detection and correction (FDC) requirements, approach, and detailed design
- Subsystem/component context diagram
- Trial code analyzed and designs modified accordingly
- Supplier documentation
    - Software Design Description(s)
    - Interface Design Description(s)
    - Updated Supplier Software Verification and Validation Plan(s)
    - Preliminary Supplier Software Test Procedure(s)
- Peer reviews for software and rework accomplished, as defined in the s/w and/or project plans
- Designs comprising the software completed, peer reviewed and placed under change control
- SRS to Computer Software Unit (CSU) level
- Updated Traceability Matrix (to CSU level)
- Verification that detailed designs cover the requirements
- Product Assurance and Software Safety plans and activities
- System safety analysis with associated verifications
- Updated HA / Software Assurance Classification Report (SACR), if necessary
- Subsystem-level and preliminary operations safety analyses
- Updated risk assessment and mitigation
- Updated reliability analyses and assessments
- Independent verification and validation (IV&V) plans and status
- Systems and subsystem certification plans and requirements (as needed)
- Configuration Management (CM) processes, including discrepancy reporting and tracking (development and post-release)
- Development environment (e.g., h/w diagram, operating system(s), compilers, DBMS, tools)
- If relevant, new compiler validated and producing acceptable object code for the target machine
- Tools needed for software implementation completed, qualified, installed and accepted, and team trained in their use
- Facilities for software implementation in place, operating, ready for use
- Build plan
- Product build-to specifications for each hardware and software configuration item, along with supporting trade-off analyses and data
- Coding, integration, and test plans and procedures
- V&V plan (including requirements and specification)
- Test team roles, functions, support required are defined

- Software Test Plan (integration and test procedure outlines)
- Test procedures
- Test levels (e.g., unit testing, integration testing, system testing) – description, who executes, test environment, standards followed, verification methodologies
- Testing preparation and execution activities, incl. testing of reused/heritage software, if applicable
- Build test timeline and ordered list of components and requirements to be tested in each build
- Test environments for each test level – diagram and description of tools, testbeds, facilities
- Test group trained prepared to evaluate the code using their facilities and tools
- Software for testing / activation
- Software requirement verification recording, monitoring, and current status – databases and test reports; sample test verification matrix
- System and acceptance testing – operational scenarios to be tested, including stress tests and recovery testing, if applicable
- Acceptance process – reviews (e.g., Acceptance Test Readiness Review, Acceptance Test Review), approval, and signoff processes
- Acceptance criteria
- Delivery, Installation, Maintenance
    - Disposition of source code and tools, handling of load images, installation of databases, etc.
    - Version identification and documentation
    - Maintenance plan, if applicable, including disposition of COTS components (source code, licenses, etc.)
    - Close-out and archive of software products
- Launch site operations plan
- Checkout and activation plan
- Disposal plan (including decommissioning or termination)
- Preliminary Operations Handbook
- Revised Draft of Programmer's Manual
- Draft of User's Manual
- Lessons Learned
    - Review of existing Lessons Learned from previous projects
    - Lessons Learned captured from software areas of the project; indicate the problem or success that generated the Lesson Learned, what the Lesson Learned was, and its applicability to future projects
    - Confirmation that Lessons Learned added to Lessons Learned database
- Status of change requests
- Software inputs / contributions to updated Project Plan

Exit/Success Criteria

- All supplier software requirements have been mapped to the software design
- All elements of the design are compliant with functional and performance requirements
    - Detailed design is expected to meet requirements with adequate margins at acceptable level of risk
- Interface control documents are sufficiently matured to proceed with fabrication, assembly, integration, and test, and plans are in place to manage any open items
- High confidence exists in the product baseline, and adequate documentation exists or will exist in a timely manner to allow proceeding with coding, integration, and test
- Product verification and product validation requirements and plans are complete
- Verification approach is viable, and will confirm compliance with all requirements

- Testing approach is comprehensive, and planning for system assembly, integration, test, and launch site and mission operations is sufficient to progress into next phase
- Adequate technical and programmatic margins and resources exist to complete development within budget, schedule, and risk constraints
- Risks to mission success are understood and credibly assessed, and plans and resources exist to effectively manage them
- Software inputs / contributions to
    - Safety and mission assurance (e.g., safety, reliability, maintainability, quality, and EEE parts) have been adequately addressed in system and operational designs, and any applicable S&MA products (e.g., Probabilistic Risk Assessment (PRA), system safety analysis and failure modes and effects analysis) have been approved
- Management processes used by the project team are sufficient to develop and operate the mission
- High priority RIDs against the SDD are closed/actions are completed and customer approval to proceed to next phase
- Approved readiness to proceed with software implementation and test activities
- SMP, software detailed designs, and unit test plans are an adequate and feasible basis for the implementation and test activities
- Products (listed above) are approved, baselined, and placed under configuration management

Production Readiness Review (PRR)

A PRR is held for FS&GS projects developing or acquiring multiple or similar systems greater than three or as determined by the project. The PRR determines the readiness of the system developers to efficiently produce the required number of systems. It ensures that the production plans; fabrication, assembly, and integration enabling products; and personnel are in place and ready to begin production. – NPR 7123.1

Entrance Criteria

- Significant production engineering problems encountered during development are resolved
- Design documentation adequate to support production
- Production plans and preparation adequate to begin fabrication
- Production-enabling products and adequate resources available, allocated, and ready to support end product production
- Production plans
- Production risks and mitigations
- Schedule

Exit/Success Criteria

- System requirements fully met in final production configuration
- Adequate measures in place to support production
- Design-for-manufacturing considerations ensure ease and efficiency of production and assembly
- Risks identified, credibly assessed, and characterized, and mitigation efforts defined
- Delivery schedules verified
- Alternate sources for resources identified, as appropriate

- Required facilities and tools are sufficient for end product production
- Specified special tools and test equipment are available in proper quantities
- Production and support staff are qualified
- Production engineering and planning are sufficiently mature for cost-effective production
- Production processes and methods are consistent with quality requirements
- Qualified suppliers are available for materials that are to be procured

System Integration Review (SIR)

Entrance Criteria

- Integration plans and procedures completed and approved
- Segments and/or components available for integration
- Mechanical and electrical interfaces verified against the interface control documentation
- All applicable functional, unit-level, subsystem, and qualification testing conducted successfully
- Integration facilities, including clean rooms, ground support equipment, electrical test equipment, and simulators ready and available
- Support personnel adequately trained
- Handling and safety requirements documented
- All known system discrepancies identified and disposed in accordance with agreed-upon plan
- All previous design review success criteria and key issues satisfied in accordance with agreed-upon plan
- Quality control organization ready to support integration effort
- V&V plans, test plans

Exit/Success Criteria

- Adequate integration plans and procedures are completed and approved for the system to be integrated
- Previous component, subsystem, and system test results form a satisfactory basis for proceeding to integration
- Risk level is identified and accepted by program/project leadership, as required
- Integration procedures and work flow have been clearly defined and documented
- Review of integration plans, as well as procedures, environment, and configuration of items to be integrated, provides a reasonable expectation that integration will proceed successfully
- Integration personnel have received appropriate training in integration and safety procedures

TEST READINESS REVIEW (TRR)

Entrance Criteria

- Objectives of testing clearly defined and documented, and test plans, procedures, environment, and configuration of test item(s) support those objectives
- Configuration of system under test defined and agreed to; all interfaces placed under configuration management or defined in accordance with an agreed-to plan, and version description document available to TRR participants
- Applicable functional, unit-level, subsystem, system, and qualification testing conducted successfully; results available
- All TRR-specific materials, such as test plans, test cases, and procedures, available to all participants prior to TRR
- Updated and current baselined documentation (from previous reviews - SRR, PDR, CDR
- Updated requirements and design documentation
- Required documents in the state/status required Deviations? Waivers?
- All known system discrepancies identified and disposed in accordance with agreed-upon plan
- All previous design review success criteria and key issues satisfied in accordance with agreed-upon plan
- All required test resources people (including a designated test director), facilities, test articles, test instrumentation, and other test enabling products identified and available to support required tests
- Facilities and tools for integration and test ready, qualified, validated, and available for operational use including test engineering products (test cases, procedures, tools, etc.), test beds, simulators, and models
- Roles and responsibilities of all test participants defined and agreed to
- Test contingency planning accomplished, and all personnel trained
- Supplier Software Version Description(s)
- Software Build from CM
- Operational software ready for testing
- Informal Dry Run completed without errors
- Outstanding Software Change Requests (SCRs) ready for review
- Updated Traceability Matrix
- All requirements included in test procedure document and uniquely identified and traceable in the SRTM
- Requirements Analysis and Traceability Reports (with possible RIDs)
- Code Analysis and Assessment Results (including SCRs, RIDs, etc.)
- Metric Data and Reports (implementation and test)
- Description of System Test Approach
- Test plan includes safety critical test scenarios
- Test plan includes test scenarios for all software/system requirements defined in the SRTM, tests that check the performance at the limits of ranges specified for the requirements and operational scenarios; includes test limitations and/or constraints
- Validation of operations and users manuals
- Test case structure established that identifies for each test:
    - Software requirements to be tested
    - Required inputs
    - Facilities and test tools required
    - Expected outputs and analysis methods
    - Software entities to be exercised by the test
- Configuration for system testing
- Summary of Quality Assurance (QA) activities used during development
- Successful audit of the VDD (such as FSW) including fixes

- Any current risks, issues, or requests for action (RFAs) that require follow-up and how they will be tracked to closure
- Results of Testing completed to date
  - Objectives of tests
  - Confirm all steps in the test runs are documented
  - Results and Safety Critical Test results
  - Tests performed
  - Successful Tests
  - Known problems
  - Deviations
  - Waivers
  - Issues
- Software Test Process
  - Build and System Test Methodology
  - Electrostatic Discharge considerations
  - Safety critical software verification considerations
  - Software test standards (including use of CM)
  - CM process and Procedures used for testing and how each was verified prior to usage
  - Process for capturing test data and storing it in the CM system
  - Test procedure red-line process
  - If/how a test can be resumed if error found during testing
  - Discrepancy Reporting System
  - Process for tracking Test Progress
  - Role of Quality Assurance including redlining and QA witnessing role and responsibilities
  - Any safety and security issues relevant to the testing activity
  - All workarounds and non-functioning software components
  - Time required for testing; include schedule and analysis of time needed on various environments / testbeds / Spacecraft
- List of all Requirements Documents relevant to Acceptance testing
- Acceptance Test Readiness
  - Process for analysis of Test Results including the division of responsibility
  - Acceptance Test testbed (environment) setup (hardware)
  - Setup and use of Simulators or other Test tools and their required qualifications
  - Limitations of the testbed (environment)
  - Tests that require hardware for verification
  - Description, at a high level, of what each test does, how long it lasts, and any special circumstances
  - IV&V report/status - if applicable
  - Preparedness for Acceptance Testing
  - Requests For Action (RFAs)
  - Decision to proceed to Acceptance Testing
- SMP updated for integration and test activities
- Updated software cost estimate, and software related expenditures collection and report by life cycle phases
- Test Schedule
  - Current system test status
  - Plans for Acceptance Test
  - Acceptance Test acceptance criteria
  - Issues and Concerns
  - Test Schedule
- Schedules for integration and test established and are reasonable based on results of unit testing
- Tests reusable for regression testing
- Expected results
- Completed evaluations (in conjunction with unit testing):
  - Verification of computations using nominal data

- Verification of computations using stress data
- Verification of output options and formats
- Exercise of executable statements in units at least once
- Test of options at branch points
- Verification of standards compliance

- Completed evaluations (in conjunction with s/w integration and test):
  - Verification of performance throughout anticipated range of operation conditions including nominal, abnormal, failure and degraded mode situations
  - Verification of performance throughout anticipated range of operating conditions as various strings of units are linked together and various modes are exercised
  - Verification of end-to-end functional flows and database linkages
  - Exercise of logic switching and executive control options at least once
- Risk analysis and risk list updated and associated risk management plan prepared
- Databases for integration and test been created and validated
- Test network showing interdependencies among test events and planned time deviations for these activities prepared
- Lessons Learned
  - Plans to capture any lessons learned from test program are documented
  - Review of existing Lessons Learned from previous projects
  - Lessons Learned captured from software areas of the project; indicate the problem or success that generated the Lesson Learned, what the Lesson Learned was, and its applicability to future projects

Exit/Success Criteria

- User-defined scenarios developed to test interactive or operator-oriented software
- Peer reviews completed for implementation and tests to be performed, as defined in the software and/or project plans
- Adequate test plans are completed and approved to proceed for the system under test
- Adequate identification and coordination of required test resources are completed
- Previous component, subsystem, and system test results form a satisfactory basis for proceeding into planned tests
- Risk level is identified and accepted by program/competency leadership as required
- Objectives of testing have been clearly defined and documented, and review of all test plans, as well as procedures, environment, and configuration of test item, provide a reasonable expectation that objectives will be met
- Test cases have been reviewed and analyzed for expected results, and results are consistent with test plans and objectives
- Test personnel have received appropriate training in test operation and safety procedures
- Provisions have been made should test levels or system response exceed established limits or if the system exceeds its expected range of response
- Software is ready to be tested
- Formal dry test run completed
- SMP, software implementations, and test are an adequate and feasible basis for integration and test activities
- Products (listed above) are approved, baselined and placed under configuration management

System Acceptance Review (SAR)

Entrance Criteria

- A preliminary agenda coordinated (nominally) prior to SAR
- Technical products made available to participants prior to SAR (noted in this list)
- Results of the SARs conducted at the major suppliers
- Transition to production and/or manufacturing plan
- Product verification results / test reports
- Product validation results
- Documentation that the delivered system complies with the established acceptance criteria
- Documentation that the system will perform properly in the expected operational environment
- Technical data package updated to include all test results
- Certification package
- Updated risk assessment and mitigation
- Successfully completed previous milestone reviews
- Remaining liens or unclosed actions and plans for closure
- Baselined Software Build
- Metrics Data and Reports
- Software presentation prepared for AR
    - Software overview
    - Project System Diagram
    - Functional software overview
    - Software products/artifacts
    - Software traceability matrix examples
    - STPr/SVVPr status
    - Open RIDs
    - Open SCRs
    - Software summary and recommendations
- Lessons Learned
    - Review of existing Lessons Learned from previous projects
    - Lessons Learned captured from software areas of the project; indicate the problem or success that generated the Lesson Learned, what the Lesson Learned was, and its applicability to future projects
    - Confirmation that Lessons Learned added to Lessons Learned database

Exit/Success Criteria

- Required tests and analyses are complete and indicate that system will perform properly in expected operational environment
- Risks are known and manageable
- Software system meets established acceptance criteria
- Required safe shipping, handling, checkout, and operational plans and procedures are complete and ready for use
- Technical data package is complete and reflects delivered system
- All applicable lessons learned for organizational improvement and system operations are captured
- Software system has sufficient technical maturity to authorize shipment to designated operational facility or launch site

Operational Readiness Review (ORR)

Entrance Criteria

- All validation testing completed
- Test failures and anomalies from validation testing resolved and results incorporated into all supporting and enabling operational products
- All operational supporting and enabling products (e.g., facilities, equipment, documents, updated databases) that are necessary for the nominal and contingency operations have been tested

and delivered/installed at the site(s) necessary to support operations
- Software user's manual completed
- Operations manual complete
- Software inputs / contributions to
  - Training provided to users and operators on correct operational procedures for system
  - Ground Systems Readiness
    - Diagram describing main functionality for project, how parts interact, and main flow of data between major functional parts
    - Problem Reporting and Change Request process for Discrepancy Reports (DR), Enhancement Reports (ER), Database Change Requests (DCR)
    - Current DR, ER, DCR status, include historical trend data, and details on current open DRs, ERs, DCRs
    - Key parts of system, their current Operational Readiness, and how verified; any issues, how they will be handled, and workarounds available including when permanent fixes will be completed
    - Key interactions with other systems, their Operational Readiness, and how verified; any issues, how they will be handled, and workarounds available including when permanent fixes will be completed
    - Software freeze plan (when software is frozen for launch, what types of fixes will be approved for implementation under a freeze, etc.) and how CCB will handle software changes or bug fixes close to launch
  - Mission Operations Center Readiness
    - MOC software readiness for all systems and how verified; any issues, how they will be handled, and workarounds available including when permanent fixes will be completed
    - Testing that was done, results, criticality of problems encountered, how problems will be resolved, and schedule for correction/verification of any fix
    - Current status of procedures that will be used by the MOC; how tested, results, and schedule for correction/verification of any fix
- Flight Software Maintenance Process Planned
  - Outstanding items that need to be completed before readiness is achieved along with scheduled date
  - Confirmation that flight software table loads and code patch testing successfully completed on all processors, including all possible on-board media (e.g., RAM, EEPROM)
- Science Planning and Processing System Readiness, as applicable
  - Diagram describing Science Data Processing products and general timelines involved

- Diagram describing Science System Context (relationship of main Mission Operations Center, Mission Planning Office, Science Validation Facility, Ground stations, interconnecting networks, and the main science data Instrument teams)
- Description of these main components in high-level detail including planning and processing functions; include any special cases for launch, in-orbit checkout, end of mission, etc.; description of testing, results, and issues done to verify and validate these components
- Summary of all testing done, results, and outstanding issues for Science Data Processing

⦿ Safety and Security Issues
- Software issues with safety, how addressed, and current status
- Software issues with security, how addressed, and current status

⦿ Simulations
- Number and main details for simulations by subsystem exercised, for example: Launch, Attitude Control System, Command & Data Handling, Communication, Flight Software, Power System Electronics, Mission Operations Center, Pre-Launch, Others deemed important for project
- Outstanding issues from Simulation testing, schedule impact, workarounds, and risks; for workarounds, when will problem/issue be permanently fixed

⦿ Contingencies and Constraints
- State of Contingency Flow Chart Book and any planned updates
- List of current constraints on system, state of database that details these constraints, and any outstanding actions that need to be taken
- Audits that were done and against what areas to verify constraints
- Operational problem escalation process
- Operational emergency notification process including telephone numbers to be called

⦿ Documentation Readiness - Status of
- Version Description Document(s); its location, and any outstanding issues
- Software User's Manual; its location, and any outstanding issues
- Software Operations Plan; its location, and any outstanding issues
- Software Maintenance Plan; its location, and any outstanding issues
- Software Retirement Plan; its location, and any outstanding issues

⦿ Lessons Learned
- Lessons Learned captured from software areas of the project; indicate the problem or success that generated the Lesson Learned, what the Lesson Learned was, and its applicability to future projects
- Confirmation that Lessons Learned added to Lessons Learned database

⦿ Work Remaining
- All launch critical work that needs to be completed before launch along with expected completion data
- RFA and RID reports generated as result of this ORR

- System, including any enabling products, is determined to be ready to be placed in operational status
- All applicable lessons learned for organizational improvement and systems operations have been captured
- All waivers and anomalies have been closed
- Systems hardware, software, personnel, and procedures are in place to support operations
- All project and support (flight and ground) h/w, s/w, and procedures are ready for operations and user documentation accurately reflects the deployed state of the entire system

Exit/Success Criteria

- Summary of status for Operational Readiness
  - Ground Systems
  - Flight Systems
  - Science Systems
  - Documentation including contingency book readiness
  - Operational support and maintenance support plans
  - Configuration control procedures
  - Waivers
  - Issues
  - Decision to proceed to Operational Readiness

Flight Readiness Review (FRR)

Entrance Criteria

- Certification received that flight operations can safely proceed with acceptable risk
- System and support elements confirmed as properly configured and ready for flight
- Interfaces compatible and function as expected
- System state supports a launch "go" decision based on go/no-go criteria
- Flight failures and anomalies from previously completed flights and reviews resolved and results incorporated into all supporting and enabling operational products.
- System configured for flight
- Tests, demonstrations, analyses, audits support flight readiness

Exit/Success Criteria

- Flight vehicle is ready for flight
- Software is deemed acceptably safe for flight (i.e., meeting the established acceptable risk criteria or documented as being accepted by the PM and Designated Governing Authority (DGA))
- Flight and ground software elements are ready to support flight and flight operations
- Interfaces are checked and found to be functional
- Open items and waivers have been examined and found to be acceptable
- Software contributions to all open safety and mission risk items have been addressed

3. Use of Commercial, Government, Legacy Software

Also known as COTS, GOTS, and MOTS

Requirements

From NPR 7150.2A:

Paragraph 2.3.1 The project shall ensure that when a COTS, GOTS, MOTS, reused, or open source software component is to be acquired or used, the following conditions are satisfied: [SWE-027]

a. The requirements that are to be met by the software component are identified.
b. The software component includes documentation to fulfill its intended purpose (e.g., usage instructions).
c. Proprietary, usage, ownership, warranty, licensing rights, and transfer rights have been addressed.
d. Future support for the software product is planned.
e. The software component is verified and validated to the same level of confidence as would be required of the developed software component.

Gateway to Web Handbook

Prototype: only the top link works (view section on web)

This handbook is interactive! Click on the links below to connect to the web features:

- View this section on Web
- Comment on this section
- View this section's tags
- Download PDF of only this section (smaller file)

Guidance

Note from NPR 7150.2A: For these types of software components consider the following:

a. Supplier agreement to deliver or escrow source code or third party maintenance agreement is in place.
b. A risk mitigation plan to cover the following cases is available:
   (1) Loss of supplier or third party support for the product.
   (2) Loss of maintenance for the product (or product version).
   (3) Loss of the product (e.g., license revoked, recall of product, etc.).
c. An Agreement that the project has access to defects discovered by the community of users has been obtained. When available, the project can consider joining a product users group to obtain this information.
d. A plan to provide adequate support is in place; the plan needs to include maintenance planning and the cost of maintenance.
e. Any documentation changes required to the software management, development, operations, or maintenance plans that are affected by the use or incorporation of COTS, GOTS, MOTS, reused, and legacy\heritage software.
f. A review of any open source software licenses by the Center Counsel.

Rationale

Note from NPR 7150.2A: The project responsible for procuring off-the-shelf software is responsible for documenting, prior to procurement, a plan for verifying and validating the off-the-shelf software to the same level of confidence that would be needed for an equivalent class of software if obtained through a "development" process. The project ensures that the COTS, GOTS, MOTS, reused and open source software components and data meet the applicable requirements in this NPR assigned to its software classification as shown in Appendix D of this NPR.

Some software (e.g. COTS, GOTS) is purchased with no direct NASA or NASA contractor software engineering involvement in software development. This requirement exists in NPR 7150.2a to mitigate the risk inherent in the acquisition of COTS, GOTS and other forms of OTS software.

Projects using this type of COTS/GOTS software must know that that the acquisition and maintenance of the software is expected to meet NASA requirements as spelled out in this section of NPR 7150.2a.

This requirement also exists in NPR 7150.2a because some software, whether purchased as COTS, GOTS or developed/modified in house, may contain open source software (OSS). If OSS exists within the project software, it can affect how the software can be used in the future, including internal/external releases or reuse of the software.

COTS/GOTS Software

COTS software are products available for purchase and use without the need to conduct development activities.

GOTS software is defined in A.10 of Appendix A, NPR 7150.2A

COTS/GOTS software can include software tools (e.g. word processor or spreadsheet applications), simulations (e.g. aeronautical and rocket simulations), and modeling tools (e.g. dynamics/thermal/electrical modeling tools).

If you are planning to use COTS/GOTS products, be sure to complete the tables under the Tools section. The purpose of these tables is to ensure that the table entries are considered in your software lifecycle decisions from software acquisition through software maintenance.

If COTS/GOTS software is used for a portion of the software solution, the software requirements pertaining to that portion should be used in the testing, verification and validation of the COTS/GOTS software. For example, if a MIL STD 1553 serial communications is the design solution for the project communications link requirements, and the COTS/GOTS software design solution is used along with the COTS/GOTS hardware design solution, then the project software requirements for the serial communications link should be used to test, verify and validate the COTS/GOTS 1553 software. Other functionality which might be present in the COTS/GOTS 1553 software may not be covered by the project requirements. This other functionality should be either disabled or determined to be safe by analysis and testing.

MOTS SOFTWARE

Modified Off-the-Shelf (MOTS) software is defined in A.18 of Appendix A in NPR 7150.2A

"*One of the quickest routes to disaster is to believe that one can safely and effectively modify a COTS/GOTS product. Sometimes it does make sense and can be justified. In general, however, the use of MOTS should be an idea of last resort*."

[Tricia Oberndorf, Carnegie Mellon Software Engineering Institute (SEI), September, 2010]

The DoD has had extensive experience in COTS/GOTS and MOTS. A Lessons Learned item, Commercial Item Acquisition: Considerations and Lessons Learned, specifically includes lessons learned from MOTS. Consider the following statements from the Lessons Learned item:

- [3.1.3] Modifying the commercial items is not the best way to bridge the gap [between DoD standards and the COTS product].
  Some programs failed because of a firm expectation that commercial items should be modified to accommodate program requirements. Like many DoD programs, one private corporation fell into the trap of modifying most of its commercial items in order to give them a unique corporate flavor. As a result of the practice, many of the corporate programs modifying commercial items experienced recurring technical problems and cost overruns. In contrast, the stakeholders of a successful DoD program made a firm decision to modify system requirements and not commercial items. The program delivered the basic capability in 90 days for 20% of the cost of a previous unsuccessful effort to build the same system. The failure of the previous effort was attributed to extensive modification of commercial items.
- Footnote 17, page 16:
  The definition of commercial item from the FAR, Part 2, allows for minor" modifications made to meet Federal Government requirements. In light of problems experienced by a large number of programs that have modified commercial items, a strong position against modification is taken here.
- [3.4.6] Extensive program testing of commercial items may be required.
  Programs often underestimate the impact of testing commercial items. Often DoD application of commercial items requires qualification and operational testing and evaluation (e.g., live-fire testing) to show that the items continue to perform as expected in unique military environments. In addition, if the commercial item has been modified, regression testing at the system level may be needed to ensure that the modification does not change the expected performance of the system. For example, some programs found that higher performance engines could outperform the airframe, while others found that faster hardware or software components could introduce timing problems or security holes. Lack of insight into the internal workings of the commercial item changes the nature of the test program. One program's ability to conduct operational test and evaluation was complicated by the fact that data normally generated during the development testing was not available for analysis by the operational test team. Another program that was using multiple commercial items found that even basic, advertised capabilities of commercial

items had to be tested before the program could begin its planned integration testing. The program's initial plans and schedules for testing commercial items underestimated the effort required by a factor of six.

LEGACY/HERITAGE CODE

The definitions of Legacy/Heritage code, and Software reuse, are in Appendix A of NPR 7150.2A.

It may be desirable to maintain legacy code largely intact due to one or more of the following factors:

- The code may have a history of successful application over many runs
- No new software errors have been found in the code in some time and it has been reliable through many years of use
- The cost of upgrading the legacy code (e.g. a new software development) may be uneconomical or unaffordable in terms of time or funding
- Upgrading the legacy code could add new software errors
- Software personnel are familiar with the legacy code
- Safety reviews have been conducted on the legacy code under similar applications

On the other hand, it may be desirable to replace legacy code due to one or more of the following factors:

- No active civil servants or contractors are familiar with the code or its operation
- One or more of the following documents are missing: architecture, requirements, traceability, design, source code, unit through integration test cases, validation results, user operational manuals, non-conformances, waivers, coding standards, or other key documents.
- Lack of conditions for the installation of the software or use of the software or software development environment
- No safety review has been done on the new code in its old or new operational environment
- The legacy code may contain open source software with questionable license terms or rights
- The source code language compilers may be years out of date or even inaccessible
- Emulators may not be available
- Maintenance responsibility unknown
- Legacy code may operate on out-of-date or unavailable operating systems
- Unknown IP, licensing, exportability constraints, if any.

Determining which path to follow (keep or replace legacy code) is largely a cost-risk analysis. Further guidance on this facet of legacy code will be provided in future iterations of this Electronic Handbook.

If the decision is made to maintain the use of the legacy code it is recommended that incremental improvements be made as the code is used. Specifically,

- Requirements should be documented if not already available
- Create verification and validation documents based on testing against any vendor documentation including user's manuals
- Start configuration management on the reused code.
- Software architecture and design should be documented if not already available
- Test cases should be documented
- Software debugging and error reporting should be documented
- Have Software Assurance and Safety personnel review the legacy code and documentation
- All documentation, test results, maintenance history, and other such documents associated with legacy code should be gathered and stored if it is anticipated the code will be reused.

One author, Michael C. Feathers, has defined legacy code as code which has no tests and proceeds to advise his readers on how to work with legacy code. See the reference link.

OPEN SOURCE SOFTWARE

Open source software is considered a form of Off-The-Shelf software. Even if most of the software on a NASA project is developed in-house, it may be found in embedded open source software within the code. It may be more efficient for a software engineer to use widely available and well tested code developed in the software community for common functions than to "reinvent the wheel".

Open source software is specifically mentioned in the SWE-027 requirement in NPR 7150-2a.

What is Open Source Software?

In general usage:

> "Open source software (OSS)" is not to be confused with other forms of inexpensive or "free" software; the intention of SWE-027 is to cover any software used in the software system which was not developed in-house. More generalized information on this subject of OSS is available from Wikipedia at:
>
> http://en.wikipedia.org/wiki/Open_source_software
>
> *Verify resources used for any Wikipedia article.*

NASA specific definition:

> In NPR 2210.1C, "Release of NASA Software" under Appendix A, definitions,
>
> A.1.1.7 "Open Source Software" means software where the recipient is free to use the software for any purpose, to make copies of the software and to distribute the copies without payment of royalties, to modify the software and to distribute the modified software without payment of royalties, to access and use the source code of the software, and to combine the software with other software in accordance with Open Source

licenses/agreements. Open source software is a subcategory of Publicly Releasable software.

Planning ahead for the inclusion of Open Source Software

Whether open source software is acquired or developed by NASA or a NASA contractor, a usage policy should be established up front to avoid any possible legal issues that may arise. This policy may be developed in conjunction with advice from the Software Release Agent (even if you do not plan to release the software) and/or your NASA center's IP Legal Counsel.

Releasing NASA code containing Open Source Software

When software is released to another NASA Center, NASA project or external organization, it is important to inform the receiving party of any licenses and restrictions under which the software is released. It is important to note that additional software required to "run" the released software is not part of the software release. For example a web application that runs under the Apache Web Server does not need to include the Apache Public License as part of the relevant licenses.

Software releases are also performed when software is submitted for Space Act Awards such as the NASA Software of the Year Award. For more information on software releases one should contact the Software Release Authority at the NASA Center at which the software is being or was developed.

There are requirements and processes associated with software releases. See **NPR 2210.1C**, *"Release of NASA Software"*, located at http://nodis3.gsfc.nasa.gov/displayDir.cfm?t=NPR&c=2210&s=1A

A cautionary item from NPR 2210.1C is worth repeating here:

> (NPR 2210.1c paragraph 3.2.2.2) If a proposed release of open source software includes the release of external open source software, care shall be taken to ensure that the pertinent license for such external open source software is acceptable. For example, at least one widely used external Open Source license does not currently include an indemnification provision and further requires that all software distributed with that external open source software be distributed under the same license terms.
>
> Therefore, except for an Approved for Interagency Release or Approved for NASA Release, both the Center Office or Project that is responsible for the software and Center Patent or IP Counsel shall review and approve any proposed distribution of open source software that includes external open source software.

Caution: open source software may itself contain other open source software!

Identifying and using high pedigree Open Source Software in NASA code

Going back to the NPR 7150.2A, requirement 2.3.1.e "The software component is verified and validated to the same level of confidence as would be required of the developed software component." To achieve this level of confidence, it is recommended that software developers use

only OSS (and COTS, GOTS MOTS, Legacy as well) which has a high pedigree, i.e. a gold standard. Such OSS will typically have the following characteristics:

- There should be a strong software development model including defined processes for:
  - Bug reporting: identification, triage, and correction
  - Code modification: review and approval to commit fixes and features to the source code
  - Testing, including thorough descriptions of test cases, test runs, and test configurations
  - Code review as part of the code modification process
  - Documentation that is detailed and updated
  - Discussion lists for questions: this may include a wiki, mailing list, or live chat site
  - Leadership which ensures that the community works in a coordinated fashion to define target functionality for each release and overall product focus.
- Usually, a high quality, established open source project will have a large number of developers, Usage of the project's product(s) will occur across multiple industries both nationally and internationally.
- Metrics (e.g., number of developers) can be used to evaluate the quality of an Open Source project via sites that track a large proportion of open source software projects (e.g., http://www.ohloh.net/). Norms, such as what constitutes a large number of developers, change as the number of Open Source projects and developers grow.
- The project should provide a listing of all open source software included, embedded, or modified within the piece of open source software.

Open source software may provide more opportunity to perform verification and validation of the software to the same level of confidence as if obtained through a "development" process. Often OSS project will provide online access to detailed development and test artifacts (as described above), which may be difficult to obtain from COTS vendors.

Procurement of software by NASA – Open Source Provisions

A cautionary item from NPR 2210.1C, "*Release of NASA Software*", is worth reiterating here:

[NPR 2210.1c paragraph 1.8.3] Open source software development, as defined in paragraph A.1.1.8 [of NPR 2210.1C], may be used as part of a NASA project only if the Office or Project that has responsibility for acquisition or development of the software supports incorporation of external open source software into software. In addition, the Office or Project responsible for the software acquisition or development shall:

a. Determine the ramifications of incorporating such external open source software during the acquisition planning process specified in NASA FAR Supplement Subpart 1807.1, Acquisition Plans; and

b. Consult with the Center Patent or IP Counsel early in the planning process (see 2.4.2.1) as the license under which the open source software was acquired may negatively impact NASA's intended use.

EMBEDDED OPEN SOURCE SOFTWARE

Embedded software applications written by/for NASA are commonly used by NASA for engineering software solutions. Embedded software is software specific to a particular application as opposed to general purpose software running on a desktop. Embedded software usually runs on custom computer hardware (“avionics”), often on a single chip.

Care must b taken when using vendor supplied board support packages (BSPs) which are typically supplied with off-the-shelf avionics systems. BSPs act as the software layer between the avionics hardware and the embedded software applications written by/for NASA. Most CPU boards have BSPs provided by the board manufacturer, or third parties working with the board manufacturer.

BSPs are hardware dependent, developed on hardware/software development tools which may not be accessible years later. Risk mitigation should include hardware specific software such as BSPs, software drivers, etc.

Many BSPs are provided by board manufacturers as binary code only, which could be an issue if the BSP supplier is not available and BSP errors are found. It is recommended that a project using BSPs maintain a configuration managed version of any BSPs with release dates and notes.

Vendor reports and user forums should be monitored from time of hardware and associated software are purchased through a reasonable time after deployment. Developers should monitor suppliers or user forums for bugs, workarounds, security changes, and other modifications to software which, if unknown, could derail a NASA project. Consider the following snippet from a user forum:

> [Manufacturer Pt. No.] motherboard embedded complex electronics contains malware
>
> Published: 2010-xx-xx
>
> A [Manufacturer] support forum identifies [manufacturer’s product] motherboards which contain harmful code. The embedded complex electronics for server management on some motherboards may contain malicious code. There is no impact on either new servers or non-Windows based servers. No further information is available regarding the malware, malware mitigation, the serial number of motherboards affected, nor the source of the original infection. [Manufacturer] will send snail mail and will call affected customers.

REFERENCES

COTS Foundations: Essential Background and Terminology

By Sally J. F. Baron, International Public Procurement Conference Proceedings, 21-23 September 2006. This paper defines COTS by giving a comprehensive history, explaining essential elements and defining terms and acronyms. It focuses on the recent history since the landmark “Perry Memo” of 1994, to current progress. Important issues such as intellectual property are also presented. The purpose of this paper is to provide a background as well as a working reference for academics and government procurement officials.

NASA Study on Flight Software Complexity

March, 2009. Commissioned by the NASA Office of Chief Engineer, Technical Excellence Program, Adam West, Program Manager, and edited by Daniel L. Dvorak, Systems and Software Division, Jet Propulsion Laboratory, this study reviews the mixed blessings of COTS, Identifying and Minimizing Incidental FSW Complexity regarding COTS, COTs integration risks, and COTS lifecycle cost risks.

The Commandments of COTS: Still in Search of the Promised Land, SEI, Carnegie Mellon

By David J. Carney and Patricia A. Oberndorf, Software Engineering Institute, Carnegie Mellon University, May 1997.This article examines current government trends toward using commercial-off-the-shelf (COTS) products. It discusses both the positive and the negative effects of these trends and suggests some high-level issues for policy makers to consider.

Decision Point: Will Using a COTS Component Help or Hinder Your DO-178B Certification Effort?

November 2003, CrossTalk - Journal of Defense Software Engineering, Timothy J. Budden, AVISTA. Avionics software developers today are continually challenged to cut costs and reduce time to market, without compromising the safety of their application. Many project leaders look to commercial off-the-shelf (COTS) software components as a possible means to reduce software development costs and development time. The requirements to "prove" software quality under Defense Order (DO)-178B may be difficult, but the opportunity demands consideration of COTS module integration where possible. Understand what is certifiable, how to get the right information from your vendor, and the importance of DO-178B traceability.

Added Sources of Costs in Maintaining COTS-Intensive Systems

June 2007, CrossTalk – Journal of Defense Software Engineering, Drs. Brad and Betsy Clark, Software Metrics, Inc. . Ten years ago, work was begun at the Center for Systems and Software Engineering at the University of Southern California to develop a cost model for commercial off-the-shelf (COTS)-based software systems. A series of interviews were

conducted to collect data to calibrate this model. A total of 25 project managers were interviewed; for eight of these projects, data was collected during the original system development and maintenance phases. A common sentiment heard from the people maintaining these systems was that they turned out to be more expensive to maintain than originally envisioned and, in fact, were more costly than a comparable custom-built system. At the same time, several people expressed frustration about the difficulty of communicating to upper management the reasons why COTS-based systems were so expensive to maintain. Anecdotal evidence from these interviews is used to discuss the added sources of maintenance cost. Three different approaches or strategies for system maintenance were observed and are summarized in this article.

Sick of COTs acronyms? Mil/Aero blog by John McHale

January, 2008. John McHale, Executive editor of Military & Aerospace Electronics magazine. This humorous blog gives a bit of history on the COTS, and other, acronyms.

COTS: Commercial Off-The-Shelf or Custom Off-The-Shelf?

June 2007, Wiley F. Livingston, Jr. P.E., Software Technology Support Center (STSC), Hill AFB. A refreshing look at the cost and complexity of customizing an otherwise OTS product

Open-source vs. proprietary software bugs: Which get squashed fastest?

This article from CNET News, September 26, 2007, looks at which software is more robust, in-house or COTS.

COTS Software Integration Cost Modeling Study

June 1997. From the University of Southern California Center for Software Engineering, performed for the USAF Electronic Systems Center, this study represents a first effort towards the goal of developing a comprehensive COTS integration cost modeling tool.

Researchers: Bugs in Open Source Software are waning

May 2008. By Jacqueline Emigh, Betanews. Developers of the Linux OS, Apache Web server, and about 250 other different open source projects have removed more than 8,500 individual bugs from their code over the past two years, according to a study released this week.

Open Source Licenses

Open Source Initiative, September, 2010. A list and some guidance for Open Source Licenses

Working Effectively with Legacy Code, by Michael C. Feathers, ISBN 0-13-117705-2

Michael Feathers starts with legacy code defined as code without tests. He introduces “Characterization testing” as an important concept and an essential tool for software

developers dealing with legacy code. This is a highly recommended book if you are a software developer or manager working with legacy code. Examples are given in C, C++, C#, Ruby and Java.

Change-out: A system of systems approach to COTS management

IEEE Xplore, Sixth International IEEE Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems (ICCBSS'07), 0-7695-2785-X/07, by Sally J. F. Baron, Ph.D, Management Consulting. This paper examines such complexity, provides a visual framework for a system of systems and the relevance and importance of change-out in general. From the Sixth International IEEE Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems (ICCBSS'07)

AIAA Guide for Managing the Use of Commercial off the Shelf (COTS) Software Components for Mission-Critical Systems (G-118-2006e) (ONLINE)

The purpose of this Guide is to assist development and maintenance projects (teams and individuals) that have to address the use of, or consideration of, COTS products within large, complex systems, including but not limited to mission critical systems. This assistance is provided by capturing a set of information about COTS products (benefits, risks, recommended practices, lifecycle activity impacts) and mission critical systems (variety of MCS, special needs for MCS, differences between MCS and other types of systems) and then providing some linkage between these topics so that various types of stakeholders can find useful information. The document should be of value to both management and technical individuals/teams. It should also be of value to teams that are dealing with non-MCS, in that the scope is not limited to only MCS.

LESSONS LEARNED WITH COTS, GOTS, MOTS, REUSED, OR OSS

The following lessons learned are taken primarily from the NASA Lessons Learned Database at the NeN portal.

1. [The following information comes from the NASA Study on Flight Software Complexity listed in the reference section of this document]

   Summary: In 2007, a relatively new organization in DoD—the Software Engineering and System Assurance Deputy Directorate—reported their findings on software issues based on approximately 40 program reviews in the preceding 2½ years [Baldwin 2007]. They found several software systemic issues that were significant contributors to poor program execution. Among the seven listed were the following on COTS:
   - Immature architectures, COTS integration, interoperability.
     Later, in partnership with the NDIA, they identified the seven top software issues that follow, drawn from a perspective of acquisition and oversight. Among the seven listed were the following on COTS:
   - Inadequate attention is given to total life cycle issues for COTS/NDI impacts on life cycle cost and risk.
     In partnership with the NDIA, they made seven corresponding top software recommendations. Among the seven listed were the following on COTS:
   - Improve and expand guidelines for addressing total life cycle COTS/NDI issues.

2. [The following information comes from the NASA Lessons Learned Repository at the NeN portal]

   Summary: The Shuttle Program selected off-the-shelf GPS and EGI units that met the requirements of the original customers. It was assumed that off-the-shelf units with proven design and performance would reduce acquisition costs and require minimal adaptation and minimal testing. However, the time, budget and resources needed to test and resolve firmware issues exceeded initial projections.

   Details: see the NASA Lessons Learned Repository at the NeN portal,, Lesson 1370: http://nen.nasa.gov/portal/site/llis/index.jsp?epi-content=LLKN_DOCUMENT_VIEWER&llknDocUrl=http%3A%2F%2Fnen.nasa.gov%2Fllis_content%2F1370.html&llknDocTitle=Lessons%20Learned%20Entry:%201370]

3. [The following information comes from the NASA Lessons Learned Repository at the NeN portal.] Summary: Lessons Learned Study Final Report for the Exploration Systems Mission Directorate; Langley Research Center; August 20, 2004 had the following comments on COTS:

   Summary: There has been an increasing interest in utilizing commercially available hardware and software as portions of space flight systems and their supporting infrastructure. Experience has shown that this is a very satisfactory approach for some items, and a major mistake for others. In general, COTS [products] should not be used as part of any critical systems because of the generally lower level of engineering and product assurance used in their manufacture and test. In those situations where COTS [software] has been applied to flight systems, such as the laptop computers utilized as control interfaces on International Space Station (ISS), the cost of modifying and testing the hardware/software to meet flight requirements has far exceeded expectations, potentially defeating the reason for selecting COTS products in the first place. In other cases, such as the Checkout Launch Control System (CLCS) project at JSC, the cost of maintaining the commercial software had not been adequately analyzed and drove the project's recurring costs outside the acceptable range.

   Recommendation: Ensure that candidate COTS products are thoroughly analyzed for technical deficiencies and life cycle cost implications before levying them on the program.

   - COTS systems have potential to reduce system costs, but only if all of their characteristics are considered beforehand and included in the planned application. (Standards)
   - COTS systems that look good on paper may not scale well to NASA needs for legitimate reasons. These include sustaining engineering/update cycle/recertification costs, scaling effects, dependence on third party services and products. Need to assure that a life-cycle cost has been considered correctly. (HQ - CLCS)

   Details: see the NASA Lessons Learned Repository at the NeN portal: http://nen.nasa.gov/llis_lib/doc/1016526main_LL_Task_Final_Report.doc

4. [The following information comes from the NASA Lessons Learned Repository at the NeN portal.]

   Summary: The purpose of the Standard Autonomous File Server (SAFS) is to provide automated management of large data files without interfering with the assets involved in the acquisition of the data. It operates as a stand-alone solution, monitoring itself, and providing an automated level of fail-over processing to enhance reliability. The successful integration of COTS products into the SAFS system has been key to its becoming accepted as a NASA standard resource for file distribution, and leading to its nomination for NASA's Software of the Year Award in 1999.

   Lessons learned: Match COTS tools to project requirements. Deciding to use a COTS product as the basis of system software design is potentially risky, but the potential benefits include quicker delivery, less cost, and more reliability in the final product. The following lessons were learned in the definition phase of the SAFS/CSAFS development.

   - Use COTS products and re-use previously developed internal products.
   - Create a prioritized list of desired COTS features.
   - Talk with local experts having experience in similar areas.
   - Conduct frequent peer and design reviews.
   - Obtain demonstration [evaluation] versions of COTS products.
   - Obtain customer references from vendors.
   - Select a product appropriately sized for your application.
   - Choose a product closely aligned with your project's requirements.
   - Select a vendor whose size will permit a working relationship.
   - Use vendor tutorials, documentation, and vendor contacts during COTS evaluation period.

Test and prototype COTS products in the lab.

The prototyping and test phase of the COTS evaluation allows problems to be identified as the system design matures. These problems can be mitigated (often with the help and cooperation of the COTS vendor) well before the field-testing phase at which time it may be too costly or impossible to retrofit a solution. The following lessons were learned in the prototyping and test phase of the SAFS/CSAFS development.

- Prototype your systems hardware and software in a lab setting as similar to the field environment as possible;
  - simulate how the product will work on various customer platforms
  - model the field operations
  - develop in stages with ongoing integration and testing
- Pass pertinent information on to your customers
- Accommodate your customers, where possible, by building in alternative options

- Don't approve all requests for additional options by customers or new projects that come on line.
- Select the best COTS components for product performance even if they are from multiple vendors.
- Consider the expansion capability of any COTS product
- Determine if the vendor's support is adequate for your requirements

Install, operate and maintain the COTS field and lab components. The following lessons were learned in the installation and operation phase of the SAFS/CSAFS development.

- Personally perform on-site installations whenever possible
- Have support/maintenance contracts for hardware and software through development, deployment, and first year of operation
- Create visual representations of system interactions where possible.
- Obtain feedback from end users
- Maintain the prototype system after deployment
- Select COTS products with the ability to do internal logging

Details: see the NASA Lessons Learned Repository at the NeN portal, Lesson 1346:

http://nen.nasa.gov/portal/site/llis/index.jsp?epi-content=LLKN_DOCUMENT_VIEWER&llknDocUrl=http%3A%2F%2Fnen.nasa.gov%2Fllis_content%2F1346.html&llknDocTitle=Lessons%20Learned%20Entry:%201346

5. [The following information comes from the NASA Lessons Learned Repository at the NeN portal]

   Summary: Shortly after the commencement of science activities on Mars, the Mars Exploration Rover (MER) lost the ability to execute any task that requested memory from the flight computer. The cause was incorrect configuration parameters in two operating system software modules that control the storage of files in system memory and flash memory. Seven recommendations cover enforcing design guidelines for COTS software, verifying assumptions about software behavior, maintaining a list of lower priority action items, testing flight software internal functions, creating a comprehensive suite of tests and automated analysis tools, providing downlinked data on system resources, and avoiding the problematic file system and complex directory structure.

   Recommendations:

   - Enforce the project-specific design guidelines for COTS software, as well as for NASA-developed software. Assure that the flight software development team reviews the basic logic and functions of commercial off-the-shelf (COTS) software, with briefings and participation by the vendor.

- Verify assumptions regarding the expected behavior of software modules. Do not use a module without detailed peer review, and assure that all design and test issues are addressed.
- Where the software development schedule forestalls completion of lower priority action items, maintain a list of incomplete items that require resolution before final configuration of the flight software.
- Place high priority on completing tests to verify the execution of flight software internal functions.
- Early in the software development process, create a comprehensive suite of tests and automated analysis tools. Ensure that reporting flight computer related resource usage is included.
- Ensure that the flight software downlinks data on system resources (such as the free system memory) so that the actual and expected behavior of the system can be compared.
- For future missions, implement a more robust version of the dosFsLib module, and/or use a different type of file system and a less complex directory structure.

Details: see the NASA Lessons Learned Repository at the NeN portal, Lesson 1483

http://nen.nasa.gov/portal/site/llis/index.jsp?epi-content=LLKN_DOCUMENT_VIEWER&llknDocUrl=http%3A%2F%2Fnen.nasa.gov%2Fllis_content%2F1483.html&llknDocTitle=Lessons%20Learned%20Entry:%201483

6. [The following information comes from the NEN Lessons Learned Repository.]

   Summary: International Space Station Lessons Learned as Applied to Exploration, KSC, July 22, 2009, had the following comments on COTS:

   [23-Lesson]: Use Commercial Off-the-Shelf Products Where Possible
   An effective strategy in the ISS program was to simplify designs by utilizing commercial off-the-shelf (COTS) hardware and software products for non-safety, non-critical applications. Application to Exploration: Use of COTS products should be encouraged whenever practical in exploration programs.

   Details: see the NASA Lessons Learned Repository at the NeN portal, at:
   http://nen.nasa.gov/llis_lib/pdf/1022932main_ISSLessonsLearnedJuly2009.pdf

7. [The following information is from Commercial Item Acquisition: Considerations and Lessons Learned July 14, 2000, Office of the Secretary of Defense,]

   Summary: This document is designed to assist DoD acquisition and supported commercial items. It provides an overview of the considerations inherent in such acquisitions and summarizes lessons learned from a wide variety of programs. Although it's written with the DoD acquirer in mind, it can provide useful information you r and assist you as we move down this increasingly important path.

   Details: see Commercial Item Acquisition: Considerations and Lessons Learned

4. Software Acquisition

Purpose

This section discusses guidance for projects implementing the NPR 7150.2A requirements addressing software acquisition, including SWE-033, SWE-037, SWE-038, SWE-045 through SWE-048, and SWE-102. This guidance is intended for all persons responsible for the software acquisition process from the planning stages through contract close-out.

Gateway to Web Handbook

Prototype: only the top link works (view section on web)

This handbook is interactive! Click on the links below to connect to the web features:

- View this section on Web
- Comment on this section
- View this section's tags
- Download PDF of only this section (smaller file)

Roles

Role	Responsibility
Project Manager	Approve procurement plan
Software Project Lead	Prepare procurement plan, monitor execution of contract
System Engineer	Conduct trade studies, engineering analyses
Contracting Officer (CO)	Prepare contracts
Contracting Officer Technical Representative (COTR)	Prepare contracts
Technical Authority	Review SOW

Planning

Before software acquisition can be carried out, a need requiring a solution must be identified. During the planning stage, various solutions to address the identified need are evaluated with the following possible options:

- In-house development/service
- Contracted development/service
- Acquire OTS product
- Use/enhance existing product/service

If the solution to the need will involve software, NPR 7150.2A applies and the acquisition planning guidance below should be applied:

1. If not already done, define the scope of the system of interest.
2. If not already done, identify the goals and objectives for the software portion of the system.
3. Identify the technical requirements (functional, operational, performance).
4. Perform "make or buy" market research/trade studies to determine if an off-the-shelf (OTS) solution exists:
    - Establish criteria (and a plan) for the studies:

        - Technical requirements
        - NPR 7150.2A classification
        - Constraints and limitations (cost, schedule, resources)
        - Use past studies, known alternatives, existing make/buy criteria
    - Conduct studies.
        - Assess potential products and technologies
        - Assess how well technical requirements are addressed
        - Assess estimated costs, including support
        - Identify risks (delivery, safety, development practices used by supplier, supplier track record, etc.)
        - Assess provider business stability, past performance, ability to meet maintenance requirements, etc.
    - Identify in-house capabilities to meet the need:
        - Assess availability of existing products which could meet the need or be modified to meet the need
        - Assess availability of qualified personnel for development or modification activities
        - Assess estimated costs (time, personnel, materials, etc.), including support
            - Use past projects as basis, where appropriate
        - Identify risks
    - Determine if solution will be custom made, an existing product, or a modified existing product.
    - Review COTS/GOTS/MOTS guidance in NPR 7150.2A handbook for additional guidance and considerations.
5. Identify any acquisition risks based on requirements and “make or buy” decisions.
6. Document analysis:
    - Expected classification of the software to be acquired
    - Availability of in-house staff and funding resources
    - Availability of the software product(s)
    - Projected licensing and support costs
    - List of potential suppliers
    - Security considerations
    - Potential risks related to supplier’s viability and past performance
7. Document solution choice and basis for that choice:
    - Estimate of in-house vs. acquisition costs (including OTS solutions and any associated costs for requirements not met by the OTS solution)
    - Comparison of cost estimates to available funding
    - Risk assessment
    - Assumptions, observations, rationale, determining factors
    - Significant issues, impacts of each option
    - If solution is in-house development/service, exit this procedure
    - If solution is to acquire product/service, continue tailoring as needed based on development under contract or purchase OTS solution
    - Other planning decisions resulting in best overall value to NASA

    - Description of chosen acquisition strategy
8. Identify stakeholders based on requirements and "make or buy" decisions:
    - Those directly concerned with, or affected by, the acquisition decision.
    - May include management, the project team, procurement, customers, end users, and suppliers.
9. Report analysis and resulting decision to appropriate stakeholders.
10. Document lessons learned for future acquisition activities.
11. Develop acquisition schedule, including solicitation, supplier selection, supplier monitoring, and product acceptance and transition to operations, as appropriate.
12. Develop acquisition plan using center-specific template.

SOLICITATION, SELECTION, AWARD

Once the planning activities for software acquisition have been completed and the decision has been made to acquire the software or software development services, a selection process needs to be followed to choose the best provider for the project. This process typically begins with development of a Statement of Work (SOW). The following recommendations should be considered as part of this process. Additionally, a SOW checklist is included in the Tools section of this guidance document.

1. Develop solicitation, including SOW:
    - Acceptance criteria
    - Solicitation constraints
    - Proper requirements from the software development perspective:
        - Software classification (from NPR 7150.2A and safety criticality (from Software Safety Litmus Test)
        - Technical requirements
        - Development standard to be followed, if any
        - Development lifecycle to be followed, or indication that developer can choose appropriate lifecycle
        - Surveillance activities (and acquirer involvement) including monitoring activities, reviews, audits, decision points, meetings, etc.
        - Management and support requirements (project management, schedule and schedule updates, configuration management, non-conformance and change tracking, risk management, metrics collection, IV&V support, required records, traceability records, electronic records and code access, V&V, etc.)
        - Requirements for maintenance, support, updates, new versions, training to be included in lifecycle and cost estimates
        - Concise task and deliverable descriptions, including delivery format
        - Media format for code deliverables
        - Templates or Data Item Descriptions (DID) for documentation deliverables
        - Complete set of deliverables with delivery dates, review periods, and acceptance procedures for each
        - Time period for responses to review findings, including making changes
        - Data Requirements Documents for deliverables, if appropriate

    - Government and contractor proprietary, usage, ownership, warranty, data, and licensing rights, including transfer
    - Requirement to include notice of use of open source software in developed code
    - OTS software requirements (identify which requirements are met by OTS software, provide OTS software documentation such as usage instructions, etc.)
    - List of all mandatory NASA software development standards and DIDs, as applicable
2. Ensure proper review of SOW before delivery to procurement/contracts official:
    - Technical Authority to ensure proper flow down of NPR 7150.2A requirements
    - Coordinate with the Safety and Mission Assurance Office to ensure all QA requirements, clauses, and intended delegations are identified and included
3. Identify potential suppliers.
4. Distribute solicitation package.
5. Evaluate proposals (typically an evaluation team), based on selection criteria, including:
    - Evaluation of how well proposed solutions meet the requirements (including interface and technology requirements, NPR 7150.2A requirements)
    - Staff available
    - Past performance
    - Software engineering and management capabilities
    - Prior expertise on similar projects
    - Available resources (facilities, hardware, software, training, etc.)
    - Etc.
6. Select supplier/contractor and document basis for selection.
7. Negotiate and finalize contract:
    - Based on SOW
    - Identify and include management reviews and meetings, such as:
        - Formal reviews, such as those found in NPR 7123.1 and NPR 7120.4
        - Technical reviews
        - Progress reviews
        - Peer reviews (see Peer Reviews and Inspection topic guidance in this handbook)
        - Software quality assurance meetings
        - System integration test and verification meetings
        - System safety meetings
        - Configuration management meetings
        - Etc.
    - Consider for inclusion in contract provisions (description of the method to be used) for verification of
        - Contractor handling of requirements changes
        - Accuracy of contractor transformation of high-level requirements into software requirements and detailed designs

    - Interface specifications between the contractor's product and systems external to it
    - Adequacy of contractor's risk management plan and its implementation in accordance with the required activities in the project Software Risk Management Plan
    - Adequacy of the contractor's integration and test plan and its implementation in accordance with the required activities in the project Software Integration and Test Plan
    - Adequacy of the contractor's configuration management plan and its implementation in accordance with the required activities in the project Software Configuration Management Plan
- Consider for inclusion in the contract the content and frequency of progress reports and metrics submissions
- Consider for inclusion in the contract identification of quality records to be maintained by the supplier
- Consider for inclusion in the contract the delivery process and how it will be accomplished; if incremental development and delivery agreed upon, state how the validation process works (e.g., incremental validation) and whether it requires integration and test with software/hardware products developed by acquirer and/or other contractors or organizations (other institutes, universities, etc.)
- Consider for inclusion in the contract a policy for maintaining the software after delivery: who is responsible for maintenance of the software, tools, testbeds, and documentation updates

MONITORING AND QUALITY ASSURANCE

Once the provider has been chosen, the acquisition process moves into a monitoring role. The following guidance should be included when establishing the process for provider monitoring and quality assurance:

1. Provide technical requirements interpretation for contractor.
2. Ensure contractor requirements documents meet original intent.
3. Evaluate contractor progress with respect to cost.
4. Periodically monitor contractor skill mix to ensure agreed-upon skills and experience levels are being provided.
5. Oversee government-furnished equipment (GFE) to ensure equipment and information provided in timely manner.
6. Periodically assess contractor processes to ensure conformance to process requirements stated in the contract (NPR 7150.2A, CMMI models at specified level):
    - Risk Management
    - Software Configuration Management
    - Software Quality Assurance
    - Software IV&V&
7. Review and assess adequacy of contractor-provided documentation and ensure contractor implementation of feedback.

- Consider using Formal Inspections.

8. Track status considering the following example questions:
   - Is the contractor meeting their staffing plan?
   - Have the project and the contractor met the user's needs?
   - Does the contractor have stable, educated staff?
   - Does the contractor's project have adequate resources (e.g., adequate staffing and computer resources)?
   - Is there realistic planning/budgeting in place?
   - Is the build plan being met?
   - Does the contractor have a good understanding of what is required?
   - Are the requirements stable?
   - Is the completion of designed functionality visible?
   - Is the evolving capability and performance of the contractor's product likely to impact development on the acquirer side of the interface?
   - Are integration and testing proceeding as planned?
   - Is contractor cost/schedule performance on target?
   - Is contractor developing a quality product?
9. Provide regular status reviews to higher-level management on contractor progress.
10. Regularly assess status of identified risks and provide reports during management reviews.

CONTRACT ADMINISTRATION

In addition to monitoring the selection provider's progress and quality, contract administration activities are also carried out for the project. The following guidance should be included when establishing the process for contract administration:

1. Regularly assess contractor financial data and invoices against budget.
2. Work with Contracting Officer to ensure timely resolution of any contract-related issues.
3. Work with Contracting Officer to ensure timely address of needed modifications to contract terms and conditions, as needed.
   - Primarily those affecting schedule, costs, services/products, resources (people, facilities), deliverables
4. Periodically evaluate contractor performance in manner consistent with contract and provide documented evaluation to Contracting Officer.

Product Acceptance and Control

Once the provider is ready to deliver the software product, the acquirer should have a process in place for review and acceptance of the product. The following guidance should be included when establishing the process for product acceptance:

1. Review deliverables based on agreed-upon acceptance criteria (or generally accepted standards if no criteria established), document results, and work with contractor to resolve acceptance issues.
   - Typically, an acceptance test plan is created addressing the following:
     - Acquirer and contractor roles and responsibility
     - Defined Test Strategy
     - Defined Test Objectives
     - Defined Acceptance Criteria
     - Developed Test Scenarios
     - Developed Test Scripts
     - Developed Test Matrix
     - Time and Resources Estimate
     - Approval Cycle
     - Strategy for post-delivery problem resolutions
   - Once approved, the test plan is executed and results are documented:
     - Select Test Tools
     - Select and Train Team Members
     - Execute the Test Plan (Manual and Automated Methods)
     - Track Test Progress
     - Regression Test
     - Document Test Results
     - Resolve Problems
2. Place formal deliverables under configuration control.
3. After acceptance of delivered products, support transition to an operational and/or maintenance environment.

Contract Close-Out

Contract close-out is the final acquisition step. The following guidance should be included when establishing the process for contract close-out:

1. Verify satisfaction of all contract terms and conditions, considering the following sample questions:
   - Has the contract period of performance expired (level of effort type contract)?
   - Have all deliverables been delivered (completion type contract)?
   - Have all Contract Data Requirements List (CDRL) Items been delivered and accepted?
   - Was the contractor's performance of the SOW acceptable?
   - If the contract involved patent rights, has the final patent report been filed?
   - Has the final invoice been received?
2. Verify return of all GFE, as appropriate.

3. Complete final reports as requested by Contracting Officer.
4. Provide final contractor performance evaluation to Contracting Officer.
5. Capture Lessons Learned, if not captured earlier in the project lifecycle.

USEFUL TOOLS

The documents below are tools collected from various Centers that work well and produce good results. They are included here as aides for carrying out the software acquisition process.

Statement of Work Checklist

This checklist was taken directly from the Langley Research Center Statement of Work (SOW) Review Procedure, LMS-CP-5523 Rev. B, and includes practices recognized by OCE as practices that work very well for NASA. See the NASA Agency PAL for the latest version of this checklist, click here for link on NEN:

https://nen.nasa.gov/web/software/nasa-software-process-asset-library-pal?p_p_id=webconnector_WAR_webconnector_INSTANCE_PA7b&p_p_lifecycle=1&p_p_state=normal&p_p_mode=view&p_p_col_id=column-2&p_p_col_count=1&_webconnector_WAR_webconnector_INSTANCE_PA7b_edu.wisc.my.webproxy.URL=https%3A%2F%2Fnx.arc.nasa.gov%2Fnx%2Fdsweb%2FGet%2FDocument-499443%2F5523_7-24-06_SOW_RevA_generic-R1V0.doc

Note: Items in gray text are provided as examples and explanatory guidance. **For additional guidance and examples on developing a Statement of Work see** URL: http://sw-eng.larc.nasa.gov/docs/statements_of_work.html and LPR 5000.2 "Procurement Initiator's Guide, Section 12 and 13.

Editorial Checklist

a. Is the SOW requirement in the form: "**Who**" shall "**Do What**"? E.g., "The Contractor shall (perform, provide, develop, test, analyze, or other verb followed by a description of what)." Example SOW requirements:
   - The Contractor shall design the XYZ flight software…
   - The Contractor shall operate the ABC ground system…
   - The Contractor shall provide maintenance on the following…
   - The Contractor shall report software metrics monthly …
   - The Contractor shall integrate the PQR instrument with the spacecraft…
b. Is the SOW requirement a simple sentence that contains only one requirement? Compound sentences that contain more than one SOW requirement need to be split into multiple simple sentences. (For example, "The Contractor shall do ABC and perform XYZ" should be rewritten as: "The Contractor shall do ABC. The Contractor shall perform XYZ.")
c. Is the SOW composed of simple, cohesive paragraphs, each covering a single topic? Paragraphs containing many requirements should be divided into sub-paragraphs for clarity.
d. Has each paragraph and subparagraph been given a unique number or letter identifier? Is the numbering / lettering correct?
e. Is the SOW requirement in the active rather than the passive voice? Passive voice leads to vague statements. (For example, state: "The Contractor shall hold monthly management review meetings…" instead of "Management review meeting shall be held monthly …")
f. Is the SOW requirement stated positively as opposed to negatively? (i.e., replace statements such as "The Contractor shall not exceed the budgetary limits specified…" with "The contractor shall comply with the budgetary limits specified...")

g. Is the SOW requirement grammatically correct?
h. Is the SOW requirement free of typographic errors, misspellings, and punctuation errors?
i. Have all acronyms been defined in an Acronym List or spelled out in the first occurrence?
j. Have the quantities, delivery schedules, and delivery method been identified for each deliverable within the SOW or a separate attachment/section?
k. Has the content of documents to be delivered been defined in a separate attachment/section and submitted with the SOW?
l. Has the file format of each electronic deliverable been defined? (e.g., Microsoft – Project, Adobe – Acrobat PDF, National Instruments – Labview VIs)

Content Checklist

a. Are correct terms used to define the requirements?
    1. **Shall** = requirement (binds the contractor)
    2. **Should** = goal (leaves decision to contractor; avoid using this word)
    3. **May** = allowable action (leaves decision to contractor; avoid using this word)
    4. **Will** = facts or declaration of intent by the Government (use only in referring to the Government)
    5. **Present tense** (e.g., "is") = descriptive text only (avoid using in requirements statements; use "shall" instead)
    6. NEVER use '**must**'
b. Is the scope of the SOW clearly defined? Is it clear what you are buying?
c. Is the flow and organizational structure of the document logical and understandable? (See LPR 5000.2 "Procurement Initiator's Guide", Section 12 for "helpful hints".) Is the text compatible with the title of the section it's under? Are sub-headings compatible with the subject matter of a heading?
d. Is the SOW requirement clear and understandable?
    1. Can the sentence only be understood one way?
    2. Will all terminology used have the same meaning to different readers without definition? Has any terminology for which this is not the case been defined in the SOW? (e.g., in a Definitions section or Glossary.)
    3. Is it free from indefinite pronouns ("this", "that", "these", "those") without clear antecedents? (e.g., replace statements such as "These shall be inspected on an annual basis." with "The fan blades shall be inspected on an annual basis.")
    4. Is it stated concisely?
e. Have all redundant requirements been removed? Redundant requirements can reduce clarity, increase ambiguity, and lead to contradictions.
f. Is the requirement consistent with other requirements in the SOW, without contradicting itself, without using the same terminology with different meanings, without using different terminology for the same thing?
g. If the SOW includes the delivery of a product (as opposed to just a services SOW):
    1. Are the technical product requirements in a separate section or attachment, apart from the activities that the contractor is required to perform? The intent is to clearly delineate between the technical product requirements and requirements for activities the contractor is to perform. (E.g., separate SOW statements "The contractor shall"

from technical product requirement statements such as “The system shall” and “The software shall”.)

2. Are references to the product and its sub-elements in the SOW at the level described in the technical product requirements?
3. Is the SOW consistent with and does it use the same terminology as the technical product requirements?

h. Is the SOW requirement free of ambiguities? Make sure the SOW requirement is free of vague terms. (For example, “as appropriate”, “any”, “either”, “etc.”, “and/or”, “support”, “necessary”, “but not limited to”, “be capable of”, “be able to”)?
i. Is the SOW requirement verifiable? Make sure the SOW requirement is free of unverifiable terms. For example, “flexible”, “easy”, “sufficient”, “safe”, “ad hoc”, “adequate”, “accommodate”, “user-friendly”, “usable”, “when required”, “if required”, “appropriate”, “fast”, “portable”, “light-weight”, “small”, “large”, “maximize”, “minimize”, “optimize”, “sufficient”, “robust”, “quickly”, “easily”, “clearly”, other “ly” words, other “ize” words.
j. Is the SOW requirement free of implementation constraints? SOW requirements should state WHAT the contractor is to do, NOT HOW they are to do it. For example, “The Contractor shall design the XYZ flight software” states WHAT the contractor is to do, while “The Contractor shall design the XYZ software using object-oriented design” states HOW the contractor is to implement the activity of designing the software. In addition, too low a level of decomposition of activities can result in specifying how the activities are to be done, rather than what activities are to be done.
k. Is the SOW requirement stated in such a way that compliance with the requirement is verifiable? Does a means exist to measure or otherwise assess its accomplishment? Can a method for verifying compliance with the requirement be defined (e.g., described in a Quality Assurance Surveillance Plan)?
l. Is the background material clearly labeled as such (i.e., included in the background section of the SOW if one is used)?
m. Are the assumptions able to be validated and restated as requirements? If not, the assumptions should be deleted from the SOW. Assumptions should be recorded in a document separate from the SOW.
n. Is the SOW complete, covering all of the work the contractor is to do?
    1. Are all of the activities necessary to develop the product included? (E.g., system, software, and hardware activities for the following: requirements, architecture, and design development; implementation and manufacturing; verification and validation; integration testing and qualification testing.)
    2. Are all safety, reliability, maintainability (e.g., mean time to restore), availability, quality assurance, and security requirements defined for the total life of the contract?
    3. Does the SOW include a requirement for the contractor to have a quality system (e.g., ISO certified), if one is needed?
    4. Are all of the necessary management and support requirements included in the SOW? (For example, project management; configuration management; systems engineering; system integration and test; risk management; interface definition and management; metrics collection, reporting, analysis and use; acceptance testing; NASA Independent Verification and Validation support tasks.)

5. Are clear Performance Standards included and sufficient to measure contractor performance? (e.g., systems, software, hardware, and service performance standards for the following: schedule, progress, size, stability, cost, resources, and defects.) See *Guidance on System and Software Metrics for Performance-Based Contracting* at: http://sw-eng.larc.nasa.gov/docs/statements_of_work.html for more information and examples on Performance Standards.
6. Are all of the necessary service activities included? (For example, transition to operations, operations, maintenance, database administration, system administration, data management.)
7. Are all of the Government surveillance activities included? (For example, project management meetings; decision points; requirements and design peer reviews for systems, software, and hardware; demonstrations; test readiness reviews; other desired meetings (e.g., Technical Interchange Meetings); collection and delivery of metrics for systems, software, hardware, and services (e.g. to provide visibility into development progress and cost); electronic access to technical and management data; access to subcontractors and other team members for the purposes of communication.)
8. Are the Government requirements for contractor inspection and testing addressed, if necessary?
9. Are the requirements for contractor support of Government acceptance activities addressed, if necessary?

o. Does the SOW only include contractor requirements? It should not include Government requirements.
p. Does the SOW give the contractor full management responsibility and hold them accountable for the end result?
q. Is the SOW sufficiently detailed to permit a realistic estimate of cost, labor, and other resources required to accomplish each activity?
r. Are all deliverables identified (e.g., status, financial, product deliverables)? The following are examples of deliverables that are sometimes overlooked: management and development plans; technical progress reports that identify current work status, problems and proposed corrective actions, and planned work; financial reports that identify costs (planned, actual, projected) by category (e.g., software, hardware, quality assurance); products (e.g., source code, Maintenance/User Manual, test equipment); and discrepancy data (e.g., defect reports, anomalies). All deliverables should be specified in a separate document except for technical deliverables which should be included in the SOW (e.g. hardware, software, prototypes, etc.).
s. Does each technical and management deliverable track to a paragraph in the SOW? Each deliverable should have a corresponding SOW requirement for its preparation (e.g., the SOW identifies the title of the deliverable in parenthesis after the task requiring the generation of the deliverable).
t. Are all reference citations complete?
    1. Is the complete number, title, and date or version of each reference specified?
    2. Does the SOW reference the standards and other compliance documents in the proper SOW paragraphs?

3. Is the correct reference document cited and is it referenced at least once?
4. Is the reference document either furnished with the SOW or available at a location identified in the SOW?
5. If the referenced standard or compliance document is only partially applicable, does the SOW explicitly and unambiguously reference the portion that is required of the contractor?

Critical and/or Complex Requirements Checklist

Note: The checklist items below may be duplicative of items included earlier in this Appendix but are summarized here to specifically identify what is required for critical and/or complex procurements.

a. Does the SOW include the name or identification of all critical and/or complex items (i.e., specifications [e.g. IEEE Standards, NFPA Standards], drawings, process requirements [e.g. LMS-CPs], inspection instructions, and other relevant technical data, as applicable)?
b. Are the requirements for design, test, examination, inspection, and related instructions for acceptance by the Government included in the SOW where applicable?
c. Are the requirements for test specimens (e.g. production method, number, storage conditions) included in the SOW if applicable? These specimens could be used by the Government for design approval, inspection, investigation or auditing.

Example Templates: The following NASA Data Item Descriptions (DIDs) are listed as sample templates for the documentation templates called for during the solicitation portion of the software acquisition process. Center Process Asset Libraries (PALs) should be consulted for DIDs and Data Requirements Documents (DRDs) relevant to a specific NASA center.

NASA-STD-2100-91

NASA DIDs are defined in the NASA-STD-2100-91 Software Documentation Standard, which is available at http://satc.gsfc.nasa.gov/assure/docstd.html. The NASA DIDs provide a format for a documentation set, including what needs to be addressed in each section.

MASTER DOCUMENTATION DATA ITEM DESCRIPTIONS

- NASA-DID-000 Software Documentation Set DID
- NASA-DID-999 Template DID

MANAGEMENT PLAN DATA ITEM DESCRIPTIONS

- NASA-DID-M000 Management Plan DID
- NASA-DID-M100 Acquisition Activities Plan DID
- NASA-DID-M200 Development Activities Plan DID
- NASA-DID-M210 Training Development Plan DID
- NASA-DID-M300 Sustaining Engineering and Operations Activities Plan DID
- NASA-DID-M400 Assurance Plan DID
- NASA-DID-M500 Risk Management Plan DID
- NASA-DID-M600 Configuration Management Plan DID
- NASA-DID-M700 Delivery and Operational Transition Plan DID

PRODUCT SPECIFICATION DATA ITEM DESCRIPTIONS

- NASA-DID-P000 Product Specification DID
- NASA-DID-P100 Concept DID
- NASA-DID-P200 Requirements DID
- NASA-DID-P300 Architectural Design DID
- NASA-DID-P400 Detailed Design DID
- NASA-DID-P410 Firmware Support Manual DID
- NASA-DID-P500 Version Description DID
- NASA-DID-P600 User's Guide DID
- NASA-DID-P700 Operational Procedures Manual DID

ASSURANCE AND TEST PROCEDURES DATA ITEM DESCRIPTIONS

- NASA-DID-A000 Assurance and Test Procedures DID
- NASA-DID-A100 Assurance Procedures DID
- NASA-DID-A200 Test Procedures DID

MANAGEMENT, ENGINEERING, AND ASSURANCE REPORTS DATA ITEM DESCRIPTIONS

- NASA-DID-R000 Management, Engineering, and Assurance Reports DID
- NASA-DID-R001 Certification Report
- NASA-DID-R002 Audit Report
- NASA-DID-R003 Inspection Report
- NASA-DID-R004 Discrepancy (NRCA) Report
- NASA-DID-R005 Engineering Change Proposal
- NASA-DID-R006 Lessons Learned Report F-10
- NASA-DID-R007 Performance/Status Reports
- NASA-DID-R008 Assurance Activity Report
- NASA-DID-R009 Test Report
- NASA-DID-R010 Waiver/Deviation Request
- NASA-DID-R011 Review Report

Center DIDs and DRDs

The following DIDs and DRDs are samples available from center PALs. Consult your own center PAL for templates relevant to work performed for your center.

Marshall Space Flight Center Templates

Available from http://spi.msfc.nasa.gov/templates.html and the individual Project Asset sections of the Marshall Space Flight Center PAL.

- Software Configuration Management Plan
- Software Test Report (STR) Template
- Unit Test Procedure Template

Goddard Space Flight Center Templates
Available from http://software.gsfc.nasa.gov/ispaindx.cfm.

- Software Management Plan/Product Plan (SMP/PP) for Class A, B, & C Software
- ISD Software Management Plan/Product Plan (SMP/PP) for Class D&E Software
- Version Description Document
- Template for the Software Quality Assurance Plan
- Configuration Management Plan Template
- Other templates in progress or not available publicly

REFERENCES

1. Software Acquisition Statement of Work Guideline, SEPG-SWACQ-PRC-1, Glenn Research Center.
    a. https://nen.nasa.gov/web/software/nasa-software-process-asset-library-pal?p_p_id=webconnector_WAR_webconnector_INSTANCE_PA7b&p_p_lifecycle=1&p_p_state=normal&p_p_mode=view&p_p_col_id=column-2&p_p_col_count=1&_webconnector_WAR_webconnector_INSTANCE_PA7b_edu.wisc.my.webproxy.URL=https%3A%2F%2Fnx.arc.nasa.gov%2Fnx%2Fdsweb%2FGet%2FDocument-499288%2FSoftware%2BAcquisition%2BSOW%2BGuideline.doc
2. Prepare Presolicitation Documents, LMS-OP-4509, Langley Research Center.
3. Statement of Work (SOW) Review Procedure, LMS-CP-5523, Langley Research Center.
4. Product Requirements Development and Management Procedure, LMS-CP-5526, Langley Research Center.
5. Process for Conducting a Make/Buy Analysis, 580-SP-075-01, Goddard Space Flight Center.
6. WBS Checklist Tool, Goddard Space Flight Center.
7. Software Supplier Agreement Management Plan, Jet Propulsion Laboratory.
    a. https://nen.nasa.gov/web/software/nasa-software-process-asset-library-pal?p_p_id=webconnector_WAR_webconnector_INSTANCE_PA7b&p_p_lifecycle=1&p_p_state=normal&p_p_mode=view&p_p_col_id=column-2&p_p_col_count=1&_webconnector_WAR_webconnector_INSTANCE_PA7b_edu.wisc.my.webproxy.URL=https%3A%2F%2Fnx.arc.nasa.gov%2Fnx%2Fdsweb%2FGet%2FDocument-499361%2FJPL%2BSoftware%2BSupplier%2BAgreement%2BManagement%2BPlan%2B%28SSAMP%29%2BTemplate%2B%2804-15-02%29.doc
8. Software Assurance: Five Essential Considerations for Acquisition Officials, Mary Linda Polydys, Stan Wisseman, STSC Crosstalk, July 2005
9. A Method for Reasoning About an Acquisition Strategy, Mary Catherine Ward, Joseph P. Elm, Software Engineering Institute (SEI), 2005
10. Software Acquisition Best Practices: 2004 Edition, Adams, Eslinger, Owens, Rich, 3rd OSD Conference on the Acquisition of Software-Intensive Systems

5. Transition of Software to a Higher Classification

PURPOSE

This document addresses guidance for projects that desire to transition software from a lower to a higher classification. This guidance is provided for Technical Authorities to provide a process for determining:

- if the transition activity is within acceptable risk boundaries and, if so,
- what strategy is needed to complete the transition to the higher classification.

Gateway to Web Handbook

Prototype: only the top link works (view section on web)

This handbook is interactive! Click on the links below to connect to the web features:

- View this section on Web
- Comment on this section
- View this section's tags
- Download PDF of only this section (smaller file)

ROLES

Role	Responsibility
Technical Authority	Reviews and approves transition strategy, waivers
Software Project Lead	Review requirements gap, document transition strategy, write waiver requests, carry out transition strategy
Original Software Author	Provide documentation, code, other artifacts, and insights into software considered for transition
Software Assurance	Ensure transition strategy is carried out

TRANSITION CATEGORIES

- Non-flight to non-flight (E->D)
- Non-flight to flight (E -> C, B, or A; D ->C, B, or A)
- Flight to Flight (C-> B or A; B->A)

Greatest risk exists for software that crosses the flight boundary and for software making large transitions, so those projects must be analyzed with the greatest care.

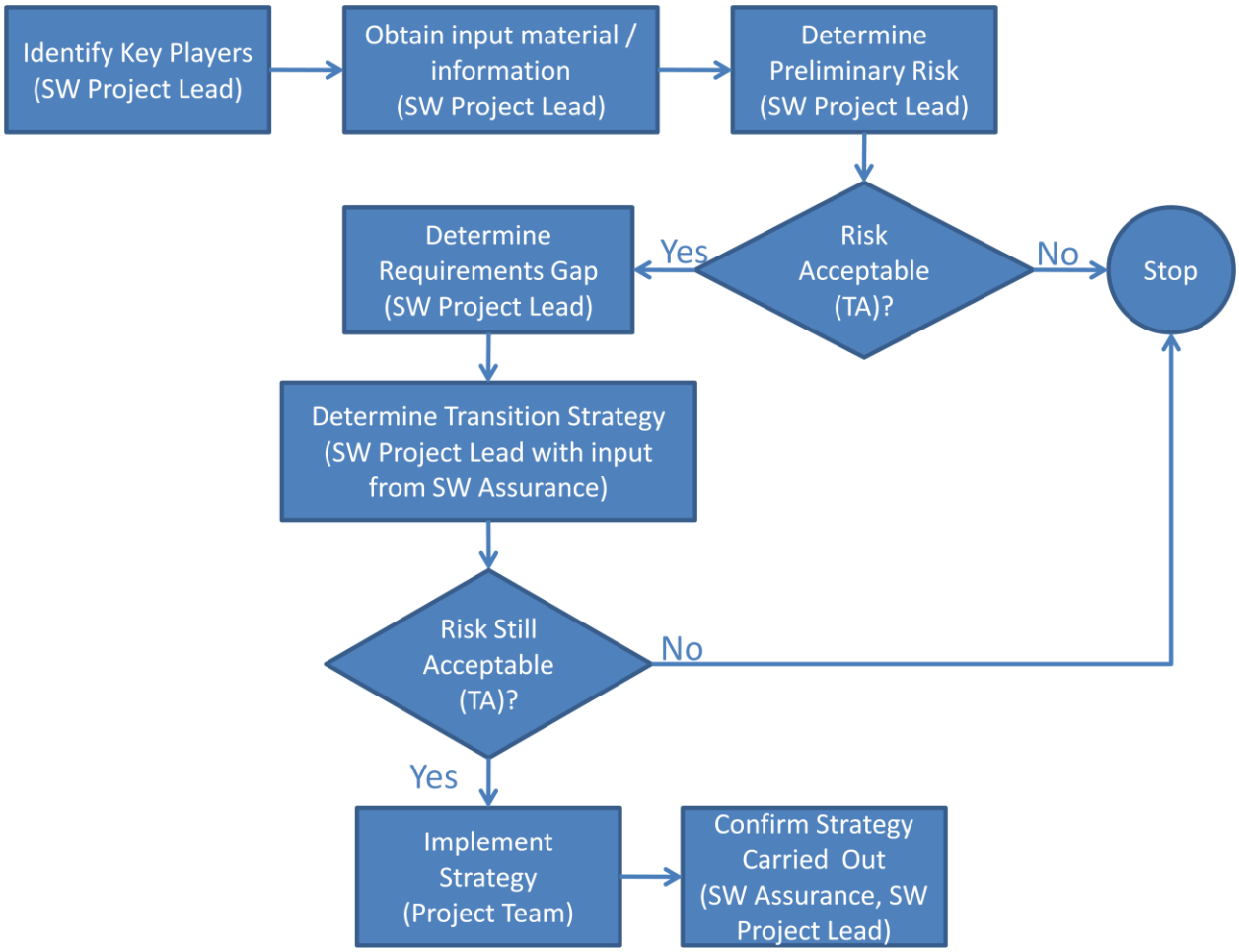
PREPARATION

Before transition risk or a strategy can be determined, the Software Project Lead should work with the original author of the software to obtain the basic information upon which that determination will be based and make it available to the Technical Authority:

- Original software proposed for transition
- Artifacts of original software development
- Description of new environment / project where transitioned software will be used

Process Overview

This process diagram includes the roles with primary responsibility for carrying out each activity or decision. While the named roles have primary responsibility for the activity, the actual completion of the activity will involve other roles as needed or in compliance with center standards.



Determine Preliminary Transition Risk

Before choosing a transition strategy, it is important for the Technical Authority to determine if the transition effort is within acceptable risk boundaries or if new development is a less risky solution. The following questions should be considered when evaluating transition versus new development solutions. Some of these topics are addressed in more detail in later analysis steps.

1. Is this a new project, midstream change of an existing project, or transition of an older, completed project?
2. What was the classification and safety criticality of the software to be transitioned? What is the higher classification and safety criticality?
    a. A larger gap, e.g., non-safety critical Class E to safety-critical Class B, means a larger number of new requirements to fulfill.

   b. The larger the gap, the more likely that the new requirements will need to be met as opposed to being tailored or waived.
   c. If the gap is too large, e.g., non-safety critical Class E to safety-critical Class A, the amount of work to close the gap may simply be too large to consider and transitioning is not a viable option.
3. Will the software operate in a different environment at the higher classification? If so, is the software appropriate to be used in the new environment? If so, what changes are needed to support this environment (hardware and software interfaces, etc.)?
4. Do software artifacts/documentation exist and can they be built upon for the transitioned software or must artifacts/documentation be created from scratch?
5. Do personnel have sufficient knowledge and skills related to the prior development to support a transition effort? What additional training is needed?
6. What are the trade-offs of coding from scratch vs. transitioning the existing work?
7. Does this preliminary assessment cause the transition to fall outside the bounds of acceptable risk which would result in abandonment of the transition?

Determine Requirements Gap

Once the Technical Authority has deemed the transition effort is within acceptable risk boundaries, the Software Project Lead has the following primary responsibilities:

- Ensure the project begins to adhere to the requirements for the higher classification and safety criticality as this will begin to reduce the existing requirements gap
- Determine the NPR 7150.2A requirements gap between the original software classification and safety criticality and the higher classification and safety criticality

The requirements gap is input to the transition strategy activity below.

To determine the requirements gap, use the NPR 7150.2A compliance matrix to determine:

1. Which requirements were met by the software when it was developed
2. Which requirements must be met for the higher classification and safety criticality
3. Which requirements that were met in the original software must be met at a more rigorous level at the higher classification
4. What is the delta number of old requirements to new ones

Determine Transition Strategy

Once the NPR 7150.2 requirements gap has been determined, a strategy for accomplishing the transition needs to be developed and documented. The Software Project Lead should develop this strategy with input from the Software Assurance organization that will be part of the group responsible for ensuring the transition strategy is carried out. Development of the transition strategy may involve a requirement by requirement review. Additionally, review and approval of the strategy and the associated risk is the responsibility of the Technical Authority.

During this process, new information may be identified that shows the transition risk is no longer acceptable. In that case, the Technical Authority can determine that transition is no longer within the acceptable risk boundaries and stop the transition attempt.

The questions below should be considered when establishing the transition strategy.

Evaluation of Additional Requirements

1. Which requirements become more significant at the higher classification? Consider:
    a. Safety requirements
    b. Risk reduction requirements
    c. Software assurance requirements
    d. Requirements for reaction to adverse conditions (data, system, environmental, etc.)
    e. Requirements for required functionality
    f. Other
2. Which requirements are candidates to be waived?
    a. What are the tradeoffs of not doing the higher level requirement(s)?
    b. What are the risks of doing/not doing the higher level requirement(s)?
    c. What requirements do not make sense to retroactively fulfill or provide little added value, e.g., due to phase of project development?
3. Which requirements will be tailored?
    a. Which requirements must be retroactively applied (e.g., peer reviews) and which requirements will be applied only to remaining work (e.g., cost estimates)?
4. Process-related requirements (e.g., configuration management, planning) should already be met, but should be checked to confirm completeness for higher level classification requirements.
    a. Are there products that need to be placed under CM that weren't for the lower class?
    b. Is the schedule and cost estimate up-to-date and detailed enough for the higher level classification?
    c. What new metrics need to be collected to meet requirements as well as to benefit future transition efforts?
    d. What processes need to be updated to meet the higher classification requirements, e.g., peer reviews, stakeholder reviews, audits)?
5. Does this effort cause the transition to fall outside the bounds of acceptable risk?

Documentation Needed

1. What is state/status of existing documentation?
    a. Review documents as well as existing change requests, inspection / peer review reports, etc. to determine state and quality of documentation.
    a. What new material/content and revisions will be necessary to meet the higher requirements?
    b. To what depth will the new content need to go to meet the higher level requirements?

2. What new documentation will be required to meet all of the higher requirements and to what depth/rigor?
3. Does this effort cause the transition to fall outside the bounds of acceptable risk?

Software Modifications Needed

1. What is the status of the software? Can it be used "as is" or do parts of the software require modification?
    a. Review existing change requests, inspection / peer review reports, test reports, etc. to determine state and quality of software.
    b. What code (to meet new requirements such as redundancy) or documentation (design, comments, etc.) must be added to fulfill the higher requirements? What is size of anticipated code modifications versus original code size?
        i. If size of modifications is significantly greater than size of original code, risk could be higher than coding from scratch.
    c. What new standards, e.g., coding standards, must be implemented / met by the code to fulfill the higher requirements?
    d. What non-essential code ("bells & whistles", test hooks, "dead" code, etc.) should be removed to conform to higher level requirements?
    e. If not already performed or data is not current, conduct static analysis to identify existing errors in the code, identify missing pieces of the code, generate complexity data, etc.
2. What is the status of the verification and validation (V&V) activities?
    a. Has V&V been performed on the software to be transitioned?
    b. Are existing V&V results invalidated by the new environment or application?
    c. What new or revised/more rigorous V&V activities (analysis, tests, results documentation, etc.) will be needed to fulfill the higher requirements?
    d. Are there areas of the project that are more safety-critical or higher risk and will require focused V&V effort at the higher level classification?
    e. If code modifications are needed, what V&V activities are required for those modifications?
3. Does this effort cause the transition to fall outside the bounds of acceptable risk?

Resources Needed

1. What resources (people, time, budget, etc.) are available for the transition effort?
    a. Are there tools and/or methods/techniques that could reduce the required effort and still provide the appropriate level of documentation? Examples:
        i. Doxygen to document design of existing code
        ii. Model-based tools to check requirements completeness
        iii. Model-based tools to check design completeness
    b. In-house personnel or contractors, i.e., are new or renegotiated contracts or internal agreements needed to support the transition effort?
2. What additional resources are needed?
    a. What additional personnel are needed, e.g., software assurance, IV&V, etc.?
    b. What new tools and/or equipment are needed, e.g., for V&V?

   c. Are there any new skills needed, e.g., new tools or coding techniques, needed to meet higher level requirements?
3. Does the resource needs cause the transition to fall outside the bounds of acceptable risk?

REFERENCES

Reference Documentation Table

This table provides guidance on what documentation a project will need to develop for software that will be reused, in whole or in part, from a previous development effort. The documentation to be developed depends on the classification of the project that will be using the software, and on decisions made by the project as to which documents are necessary based on NASA center procedures.

Software Documentation	Class A OR Class A Safety Critical	Class B OR Class B and Safety Critical	Classes C thru E and Safety Critical	Class C and NOT Safety Critical	Class D and NOT Safety Critical	Class F (In-house)	Class G (In-house)
Software Management Plan	1	1	1	1	1	1	1
Software Configuration Management Plan	1	1	1	1	1	1	1
Software Test Plan	1	1	1	1	1	1	1
Software Maintenance Plan	1	1	1			1	1
Software Assurance Plan	1	1	1	1		1	1
Software Safety Plan	1	1	1	1	1	1	1
Software Requirements Specification	2, 3, or 4	2, 3, or 4	2, 3, or 4	2, 3, or 4	2, 3, or 4	2, 3, or 4	2, 3, or 4
Software Data Dictionary	2, 3, or 4	2, 3, or 4	2, 3, or 4	2, 3, or 4		2, 3, or 4	2, 3, or 4
Software Design Description	2, 3, or 4	2, 3, or 4	2, 3, or 4	2, 3, or 4	2, 3, or 4	2, 3, or 4	2, 3, or 4
Interface Design Description	2, 3, or 4	2, 3, or 4	2, 3, or 4	2, 3, or 4		2, 3, or 4	2, 3, or 4
Software Change Request/Problem Report	1	1	1	1	1	1	1
Software Test Procedures	2	2	2	2		2	2
Software Users Manual	2, 3, or 4	2, 3, or 4	2, 3, or 4			2, 3, or 4	2, 3, or 4
Software Version Description	2, 3, or 4	2, 3, or 4	2, 3, or 4	2, 3, or 4	2, 3, or 4	2, 3, or 4	2, 3, or 4
Software Metrics Report	1	1	1	1		1	1
Software Test Report	2	2	2	2		2	2
Software Inspection/Peer Review Report	1	1	1	1		1	1

Key:
1. Project already responsible for developing this document; it needs to address integration of the transitioning software
2. Develop the document if it does not exist
3. Modify the project/document to accommodate the transitioning software
4. Incorporate existing document into the project

Note: A blank space indicates that the document does not need to be developed.

Classes E and H do not appear in the table since they are the lowest classifications. It is not possible to transition up to either of them. Documents with multiple keys allow for decisions to be made in the best interest of the project. A block with 2, 3 or 4 in it gives the project three options. For instance, a requirements document can be developed as a standalone document if it does not exist or the requirements can be included in another of the project's requirements documents or, if the requirements document exists, it can be directly incorporated into the project.

Reference Code Size Decision Chart

Decision making for software transition involves detailed investigation of the subject code and of the resources available to apply to its modification or rebuilding. The following chart provides some guidance with respect to code size, noting that Mod Size should only reflect modification within the original code, not any code to which the original software will be linked. This table on the next page should be tailored even further depending on the nature of the code's language, local experience with the programming languages involved, and availability of alternate language development resources.

	Size of Existing Code	
Size of Modifications	Small	Big
Small	Determine size of original code plus the modifications. If the determined size is big, write new code from scratch, following Center procedures and project plans. Exit this process. If the size is small then: Analyze and determine risk based on: • Relative size of original code to mod size, • Comparison of Specialized resources needed/available: • Reuse: reduced effort, but older language skills/maintenance required. • Re-code: increased effort, but newer language skills/capabilities avail. • State of the original code: (less of these items requires more resource to validate original code and assure to higher control level) • Documentation available • Known reliability/applicability/readability Evaluate the risk. If the risk is acceptable, reuse original code. Perform mods if needed. Follow Center procedures and project plans. Continue with this process. If the risk is not acceptable, write new code from scratch, following Center procedures and project plans. Exit this process.	Reuse original code. Perform mods if needed. Follow GLPR-7150.1 and project plans. Continue with this process.
Big	Write new code from scratch following Center procedures and project plans. Exit this process.	Analyze and determine risk based on: • Relative size of original code to mod size, • Comparison of Specialized resources needed/available: • Reuse: reduced effort, but older language skills/maintenance required. • Re-code: increased effort, but newer language skills/capabilities are available. • State of the original code (if few of these items exist, more resources required to validate original code and assure to higher classification level): • Documentation available • Known reliability/applicability/readability Evaluate the risk. If the risk is acceptable, reuse original code. Perform mods if needed. Follow Center procedures and project plans. Continue with this process. If the risk is not acceptable, write new code from scratch. Exit this process.

6. Validation Planning - SWE-029

Requirements

2.4.2 The project shall plan the software validation activities, methods, environments, and criteria for the project.

Notes

Software validation is a software engineering activity that shows confirmation that the software product, as provided (or as it will be provided), fulfills its intended use in its intended environment. In other words, validation ensures that "you built the right thing." Examples of validation methods include but are not limited to: formal reviews, prototype demonstrations, functional demonstrations, software testing, software peer reviews/inspections of software product component, behavior in a simulated environment, acceptance testing against mathematical models, analyses, and operational environment demonstrations. Refer to the software plan requirements for software validation planning and incorporation (NPR 7150.2A, Chapter 5).

Implementation Notes from Appendix D

Class D non-Safety Critical and Class G are labeled with "P(Center)". This means that local requirements or procedures describe validation planning sufficiently to meet the intent of this requirement.

Applicability Across Classes

This requirement applies to all classes and safety criticalities except:

- Class E and not Safety Critical
- Class H

Rationale

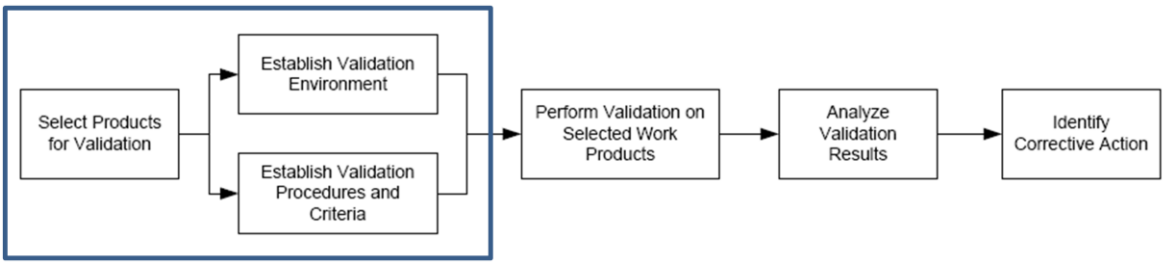
Planning should be applied to any activity that is to be repeated, that needs to be verified before use, and that requires thought before implementation. Planning the requirements validation activity allows the project team to put more thought into tasks, methods, environments, and related criteria before they are implemented. Planning also allows a current project to improve based on lessons learned from previous projects, including using more appropriate or efficient techniques and ensuring the completeness of all steps in the process.

Having a plan also allows the requirements validation activity to be reviewed, improved, and verified before it is implemented to ensure the outcome will meet the expectations and goals of

the validation activity. Planning also helps to ensure the validation activity is cost-efficient and timely.

GUIDANCE

The basic validation process is shown below with the steps addressed by this requirement highlighted:



Validation activities should not be performed in an ad hoc manner, but should be planned and captured in a validation plan document. The validation plan is typically part of a verification and validation (V&V) plan, a software V&V plan (SVVP), or is included in the Software Management / Development Plan (SMP/SDP).

The plan should cover the validation activities that will occur at various times in the development lifecycle including:

- During requirements development, validation is accomplished by bringing in the customer and outside people for a review of the requirements, e.g., focus groups, requirements reviews, etc.
- During design, validation occurs when the customers have a chance to view prototypes of the product or pieces of the product, e.g., focus groups, user groups, etc.
- During implementation, validation occurs when team members review software components for adherence to requirements, e.g., peer reviews/inspections.
- Prior to delivery, validation occurs when customers see the completed product function in a nearly operational environment, e.g., acceptance testing, operational demonstrations, etc.
- During product use, validation occurs when the product is used in the operational environment in the way the customer expects it to be used.

The project team should review the plan and validation results at various lifecycle reviews, particularly whenever requirements change throughout the duration of the project. Any identified issues should be captured in problem reports / change requests / action items and resolved before the requirements are used as the basis for development activities.

The validation plan will address more than just validation of software requirements. It should include a schedule, especially if stakeholder reviews are required to complete the validation activities and gain agreement that the requirements are a correct and acceptable description of the system or software to be implemented. Other elements to include in the overall plan:

- Scope
- Approach
- Resources
- Specific tasks and activities
- Validation methods and criteria (SWE-102)
- Identification of work products to be validated (SWE-102)
- Identification of where validation records and corrective actions will be captured (SWE-102)

The Scope and Approach sections of the plan should identify the project and define the purpose and goals of the plan including responsibilities, assumptions, and a summary of the efforts described in the plan.

Resources include personnel, environments such as simulators, facilities, tools, etc. and should include any skills and/or training necessary for those resources to carry out the validation activities.

When developing the validation plan, consider the following for inclusion:

- Identifying the key functions and/or components that require validation (based on criticality, safety, security, etc.)
- Identifying the validation methods, techniques, tests to carry out the validation activities for components as well as the system as a whole (see SWE-055)
- Identifying criteria by which success will be measured for each validation activity
- Establishing the target environment (which could be a high fidelity simulation) for validating the software or system, including validation of tools used in those environments
- Identifying how the results will be documented and reported, when and to whom they will be reported (see SWE-031)
- Issue resolution (capture and tracking to closure) for issues or findings identified during validation activities (could be as simple as using the project configuration management process) (see SWE-031)
- Identifying validation activities, as applicable, to occur during the various lifecycle phases
- Re-validation plans to accommodate changes as the system is developed
- Method for obtaining customer approval of the validation plan, if applicable

If not part of the team developing the validation plan, Software Assurance should be part of the plan's review team to ensure the plan meets all assurance requirements.

See also related requirements in this handbook:

SWE-031	Validation results
SWE-055	Requirements validation
SWE-102	Software development/management plan

SMALL PROJECTS

There is no information applicable to this section.

RESOURCES

1. Software Management Plan / Product Plan (SMP/PP) For Class A, B & C Software (Verification and Validation section), 580-TM-033-02, GSFC, http://software.gsfc.nasa.gov/AssetsApproved/PA1.2.6.1.doc
2. IEEE Standard for Software Verification and Validation. Chapter 7., IEEE Std 1012-2004
3. IEEE Guide for Software Verification and Validation Plans. IEEE Std 1059-1993
4. How to Develop A Software Validation Plan, O'Keeffe, August 2010
5. FSW Testbed Validation Description, 582-2008-006, Version 1, GSFC, 2008
6. The CMMi easy button presentation of CMMi – Validation (VAL), Software Quality Assurance.org
7. Reference Information for the Software Verification and Validation Process, NIST Special Publication 500-234, 1996

TOOLS

Software Verification and Validation Plan (SVVP) Template (based on IEEE standards), Texas State University Computer Science Department, 2001

LESSONS LEARNED

There are currently no Lessons Learned listed for this requirement.

7. Acquisition vs. Development Assessment - SWE-033

REQUIREMENTS

2.5.2 The project shall assess options for software acquisition versus development.

Notes

The assessment can include risk, cost, and benefits criteria for each of the options listed below:

- Acquire an off-the-shelf software product that satisfies the requirement.
- Develop the software product or obtain the software service internally.
- Develop the software product or obtain the software service through contract.
- Enhance an existing software product or service.

Risks are considered in software make/buy and acquisition decisions. The project needs to ensure that software products used in the design or support of human space flight components or systems include a level of rigor in risk mitigation as a software management requirement, regardless of software classification. The level of documentation needed for risk identification and tracking is defined by the Center processes.

Implementation Notes from Appendix D

NPR 7150.2A does not include any notes for this requirement.

Applicability Across Classes

This requirement applies to all classes and safety criticalities except:

- Class E and not Safety Critical
- Class H

RATIONALE

When making any decision, it is important to assess the options available in order to obtain the greatest value and benefit. Software development is no different. Choices need to be assessed in order to identify the best use of available resources (budget, time, personnel, etc.) to address a defined and scoped need while providing the greatest benefit with the least risk to the project.

GUIDANCE

When assessing solutions for software acquisition versus development, there are four possible options:

- Acquire an off-the-shelf software product that satisfies the requirement.

- Develop the software product or obtain the software service internally.
- Develop the software product or obtain the software service through contract.
- Enhance an existing software product or service.

Each option has its own benefits, costs, and risks which should be identified through market studies (for off-the-shelf products), internal assessments (for existing products), and cost-benefit analyses.

The team should assess existing software products, whether off-the-shelf or in-house, to identify how well they meet the need of the current project and whether they are suitable for the intended environment. The following information should be weighed against the defined need, architecture, environment, requirements, safety classification, budget, etc. of the current project:

- Features/functionality/capabilities
- Documentation
- Test results
- Performance record
- Safety record
- Licensing, maintenance, and support costs
- Any other relevant information

The project responsible for procuring off-the-shelf software is responsible for documenting, prior to procurement, a plan for verifying and validating the off-the-shelf software to the same level of confidence that would be needed for an equivalent class of software if obtained through a "development" process. For more detail, see SWE-027.

For development, whether internal or external, consider the following information:

- Personnel skill sets, experience, availability
- Cost associated with training, tools, post-development maintenance and support
- Company reputation, track record, history, etc. (for contracted development)
- Overall cost of development
- Intellectual Property rights
- Cost and availability of workforce should follow-on work be required
- Insight into development processes
- Schedule associated with procurement (sole source, competitive, task order, etc.) for procured software or a contracted development

Identify risks associated with each assessed option, including:

- Technical risks
- Supplier risks, including track record and support risks
- Cost and schedule risks

The team should document the results of the analysis as well as the raw data that was collected and evaluated to arrive at the final solution.

Involve the right stakeholders in the assessment process to benefit from their experience and ensure all key information is considered. Consider the following, as applicable:

- Technical personnel
- Management
- Contracts
- Procurement
- End users
- Customers
- Technical Authority

See the Acquisition Guidance topic in this handbook for additional guidance on this topic. The references in this topic may also provide additional guidance on assessing acquisition versus development options.

Additionally, center procedures addressing decision analysis and resolution may be helpful in planning and carrying out the assessment and selection process.

Small Projects

While assessing all available options is important for any software development project, it may be even more important for projects with limited budgets, personnel, or both. Small projects need to evaluate their available resources against the possible solutions to find the best fit with the least risk.

Use of existing trade studies and market analyses may reduce the cost and time of assessing available options.

RESOURCES

- Process for Conducting a Make/Buy Analysis, 580-SP-075-01, Goddard Space Flight Center (GSFC), http://software.gsfc.nasa.gov/AssetsApproved/PA2.1.1.1.doc
- ISD Decision Analysis and Resolution, 580-SP-038-001, GSFC, http://software.gsfc.nasa.gov/AssetsApproved/PA2.1.1.pdf
- See Acqusition Guidance section in this document.

Tools

Checklists of questions to ask when assessing acquisition versus development can be found in SWE-027 of this handbook

LESSONS LEARNED

Several lessons learned from the NASA Lessons Learned database (http://www.nasa.gov/offices/oce/llis/1370.html) address topics that should be kept in mind when

assessing software acquisitions versus development. While many of these lessons seem hardware-oriented, some of these lessons can also be applied to software:

1. Talk to those that have used the product before. *Outside consultants, who do not have a stake in the choice of a particular unit, should be used. Such consultants have "hands on experience" … and can be an important information source concerning their design, integration and use. Consultants who have participated in previous integrations will have knowledge of problems that other users have encountered. Consultants and other users can also provide valuable insight into the rationale and requirements that governed the original design of the unit. This information is invaluable … for identifying technical, cost and schedule risks associated with a particular … unit ...*

2. "Plug And Play" versus development. *The fact that a unit is in mass production and is a proven product does not mean that its integration into a different vehicle will be a simple, problem free "plug and play" project. A difference in application (such as aviation versus space flight) will result in the manifestation of firmware issues that may not have appeared in the original application. Unique data interfaces used by manned and some unmanned spacecraft avionics may require modification of the unit. Power supply changes and radiation hardening may also have to be performed.* While this lesson describes hardware acquisitions, software acquisitions should also keep this lesson in mind because projects have differences that can affect the suitability of software for a particular application.

3. Pay attention to "Technical Risk". *Project management may focus mainly on risk to cost and schedule, with little attention paid to technical risk. GPS project management kept Shuttle Program management well aware of the nature of a "success oriented" approach and that cost and schedule could be impacted. Analysis at the start of a project should be conducted to determine risk to cost and schedule based on the technology level, the maturity of the technology and the difference between the planned application and the application for which the box was designed originally. Software complexity should also be examined. Failure to account for technical risk can lead to cost and schedule problems*.

   An additional risk in using "off the shelf" units concerns the availability of the vendor. Can a user continue to use and maintain a product if the vendor goes out of business or stops producing and supporting the product?

4. Provide guidelines for COTS and "Faster-Better-Cheaper" implementation. A key lesson from unmanned spacecraft failures and DoD software programs is that one must understand how to properly use COTS products and apply "faster-better-cheaper" principles.

   Some projects have failed since management was not given guidance concerning how to implement a faster-better-cheaper approach. "Faster" and "cheaper" are easily understood, but "better" is difficult to define. This has also led to inconsistent application of faster-better-cheaper principles from one project to another.

A COTS policy is needed to help prevent cost, schedule and technical difficulties from imperiling projects that use COTS. Criteria for determining whether a COTS approach can be taken must be determined. Of prime importance is defining the level of insight needed into vendor software, software maintenance and certification processes.

Problems in COTS projects can arise when requirements are levied on the product that the vendor did not originally intend for the unit to meet. Using COTS may mean either compromising requirements on the COTS unit or on the integrated system. Whether or not new requirements have to be applied to the unit is a critical decision. Unfortunately, new requirements may not be recognized until the COTS product experiences difficulties in the testing and integration phases of the project.

The Shuttle Program created COTS/MOTS software guidelines for varying levels of application criticality. This recommended policy defines what considerations should be made before deciding to procure a COTS/MOTS product. The following should be examined based on the criticality (impact of failure on safety of flight or mission success) of the application and product in question:

- **Certification Plan** – How much of the vendors in-house certification can be relied upon? For critical applications, additional testing will be needed if access to test results, source code and requirements documents is not allowed. Can the unit be certified to a level commensurate with the criticality of the application?

- **Vendor Support** – This should cover the certification process and the system life cycle. The level of support should be defined based on the criticality of the system.

- **Product Reliability** – Vendor development and certification processes for both hardware and software should be examined.

- **Trade Studies** – Define “must meet,” “highly desirable” and “nice to have” requirements. Ability of the unit to meet those requirements, and at what cost, will be a major deciding factor in the COTS decision. Identify loss of operational and upgrade flexibility as well technical risks and cost associated with the product. Examine the impact of the product on the integrated system, including hardware and software interface changes. Compare the proposed COTS products to a custom developed product. Assess life expectancy of the product and its track record in the market place.

- **Risk Mitigation** – Identify areas that increase risk, such as lack of support if the vendor goes out of business or the product is no longer produced. Ensuring vendor support over the product life cycle can mitigate risk, along with gaining access to source code, design requirements, verification plans and test results. Off-line simulations of the product should also be considered. Can access be obtained to vendor information on product issues discovered by other users?

Trade studies and risk identification must be performed before committing to the use of a particular unit and integration architecture.