NASA Procedural Requirements

NPR 7150.2A
Effective Date: November 19, 2009
Expiration Date: November 19, 2014

COMPLIANCE IS MANDATORY

NASA Software Engineering Requirements

Responsible Office: Office of the Chief Engineer

Table Of Contents

List Of Figures

Preface

P.1 Purpose

Software engineering is a core capability and a key enabling technology for NASA's missions and supporting infrastructure. This NASA Procedural Requirements (NPR) supports the implementation of the NASA Policy Directive (NPD) 7120.4, NASA Engineering and Program/Project Management Policy. This NPR provides the minimal set of requirements established by the Agency for software acquisition, development, maintenance, retirement, operations, and management. This NPR is intended to support NASA programs and projects to accomplish their planned goals (e.g., mission success, safety, schedule, and budget) while satisfying their specified requirements. This NPR provides a set of software engineering requirements in generic terms to be applied throughout NASA and its contractor community. For this NPR, Software Engineering is defined as the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software: that is, the application of engineering to software. For this NPR, Software is defined as the computer programs, procedures, scripts, rules, and associated documentation and data pertaining to the development and operation of a computer system. Software includes programs and data. This definition includes commercial-off-the-shelf (COTS) software, government-off-the-shelf (GOTS) software, modified-off-the-shelf (MOTS) software, reused software, auto generated code, embedded software, firmware, and open source software components.

P.2 Applicability and Scope

P.2.1 The requirements of this NPR cover software created or acquired by or for NASA. Requirements in this NPR apply to all of the Agency's investment areas containing software systems and subsystems. The applicability of the requirements in this NPR to specific systems and subsystems within the Agency's investment areas, programs, and projects is determined through the use of the NASA-wide definition of software classes in Appendix E, in conjunction with the Requirements Mapping Matrix in Appendix D. Some projects may contain multiple systems and subsystems having different software classes. Using the Requirements Mapping Matrix, the applicable requirements and their associated rigor are adapted according to the classification and safety-criticality of the software.

P.2.2 This NPR is applicable to NASA Headquarters and NASA Centers, including Component Facilities, Technical Support Centers, and Service Support Centers. This NPR applies to the Jet Propulsion Laboratory, other contractors, grant recipients, or parties to agreements only to the extent specified or referenced in the appropriate contracts, grants, or agreements.

Note: This statement alone is not sufficient to stipulate requirements for the contractor or grant recipient. NASA project managers stipulate which of the NPR requirements are applicable to the contracts, grants, or agreements. The contract, grant, or agreement must state the requirement(s) from the NPR that apply.

P.2.3 This NPR shall be applied to software development, maintenance, retirement, operations, management, acquisition, and assurance activities started after its initial date of issuance [SWE-001]. Note: This document is not retroactively applicable to software development, maintenance, operations, management, acquisition, and assurance activities started before September 27, 2004 (i.e., existing systems and subsystems containing software for Shuttle,

International Space Station, Hubble, Chandra, etc.).

P.2.4 This NPR provides procedural requirements to the responsible NASA project managers and contracting officers for NASA contracts. The NPR is made applicable to contractors through contract clauses, specifications, or statements of work in conformance with the NASA Federal Acquisition Regulation (FAR) Supplement.

P.2.5 This NPR does not supersede more stringent requirements imposed by individual NASA organizations and other Federal Government agencies. Requirements in this NPR are identified by "shall" and a requirement number. Any material not identified by a "shall" in this NPR is informative in nature (e.g., notes, introductory text, etc.). The statements "must," "should," "are to be," "will," and "required" do not denote mandatory compliance.

P.3 Authority

a. 40 U.S.C § 11101 et seq., Clinger-Cohen Act of 1996 (Pub. L. 104-106, Division E).

b. 44 U.S.C. § 3501 et seq., Paperwork Reduction Act of 1995 (Public Law 104-13).

c. OMB Circular A-130, Transmittal Memorandum #4, Management of Federal Information Resources (11/28/2000).

P.4 Applicable Documents

The latest versions of the following documents are to be applied:

NPD 1000.0, NASA Governance and Strategic Management Handbook.

NPD 1000.3, The NASA Organization.

NPD 1000.5, Policy for NASA Acquisition.

NPD 2091.1, Inventions Made by Government Employees.

NPD 2190.1, NASA Export Control Program.

NPD 2800.1, Managing Information Technology.

NPD 2810.1, NASA Information Security Policy.

NPD 7120.4, NASA Engineering and Program/Project Management Policy.

NPD 8700.1, NASA Policy for Safety and Mission Success.

NPD 9250.1, NASA's Property Plant and Equipment Policy.

NPR 2190.1, NASA Export Control Program.

NPR 2210.1, External Release of NASA Software.

NPR 2800.1, Managing Information Technology.

NPR 2810.1, Security of Information Technology.

NPR 7120.5, NASA Space Flight Program and Project Management Requirements.

NPR 7120.6, Lessons Learned Process.

NPR 7120.7, NASA Information Technology and Institutional Infrastructure Program and Project Management Requirements.

NPR 7120.8, NASA Research and Technology Program and Project Management Requirements.

NPR 7123.1, NASA Systems Engineering Processes and Requirements.

NPR 8000.4, Agency Risk Management Procedural Requirements.

NPR 8705.2, Human-Rating Requirements for Space Systems.

NPR 8715.3, NASA General Safety Program Requirements.

NPR 8735.2, Management of Government Quality Assurance Functions for NASA Contracts.

NPR 8800.15, Real Estate Management Program Implementation Manual.

NASA-STD-8739.8, NASA Software Assurance Standard.

NASA-STD-8719.13, Software Safety Standard.

See Appendix C for a complete list of reference documents.

P.5 Measurement/Verification

Compliance with the requirements contained in this NPR is documented in accordance with paragraph 6.3.2. Engineering Technical Authorities keep records of compliance matrices, waivers, and deviations with respect to this NPR in accordance with paragraph 6.3.7. Additionally, the NASA Headquarters Office of the Chief Engineer authorizes periodic appraisals against select requirements in this NPR in accordance with paragraph 6.3.8.

P.6 Cancellation

NPR 7150.2, NASA Software Engineering Requirements, dated September 27, 2004.

NASA Chief Engineer

Chapter 1: Introduction

1.1 Overview

1.1.1 This NPR imposes requirements on procedures, design considerations, activities, and tasks used to acquire, develop, assure, and maintain software created and acquired by or for NASA programs. This NPR is designed to be a minimum set of requirements to protect the Agency's investment in software engineering products and to fulfill its responsibility to the citizens of the United States of America.

1.1.2 The requirements in this NPR have been extracted from industry standards and proven NASA experience in software engineering. Centers and software developers will find that many of the requirements are satisfied through programs, procedures, and processes that are in place.

1.1.3 The Agency makes significant investments in software engineering to support the Agency's investment areas: Space Flight, Research and Technology, Information Technology, and Institutional Infrastructure. NASA ensures that programs, projects, systems, and subsystems that use software follow a standard set of requirements. One of the goals of this NPR is to bring the Agency's engineering community together to optimize resources and talents across Center boundaries. For engineers to effectively communicate and work seamlessly among Centers, a common framework of generic requirements is needed. This NPR fulfills this need for the Agency within the discipline of software engineering.

1.1.4 This NPR makes no recommendation for a specific software life-cycle model. Each has its strengths and weaknesses, and no one model is best for every situation. Projects can evaluate the potential life-cycle models and select one that best matches the project's requirements and supports the products that are being produced. Standards or organizational policy may dictate a particular life-cycle model.

1.1.5 The NASA Headquarters Office of the Chief Engineer is committed to instituting and updating these requirements to meet the Agency's current and future challenges in software engineering. Successful experiences will be codified in updated versions of this NPR after experience has been gained through its use within the NASA software community, the collection of lessons learned from projects, and the implementation records of Engineering Technical Authority.

1.2 Organizational Capabilities and Improvement

Software engineering is a core capability and a key enabling technology necessary for the support of NASA's Mission Directorates. Ensuring the quality, safety, and reliability of NASA software is of paramount importance in achieving mission success. This chapter describes the requirements to help NASA maintain and advance organizational capability in software engineering practices to effectively meet scientific and technological objectives of the Agency.

1.2.1 The NASA Headquarters Office of the Chief Engineer shall lead, maintain, and fund a NASA Software Engineering Initiative to advance software engineering practices. [SWE-002]

1.2.2 Each Center shall maintain, staff, and implement a plan to continually advance its in-house software engineering capability and monitor the software engineering capability of NASA's contractors, as per NASA's Software Engineering Initiative Improvement Plan. [SWE-003]

Note: The requirement for the content of each Center Software Engineering Improvement Plan is defined in Chapter 5. Each Center has a current Center Software Engineering Improvement Plan on file in the NASA Headquarters' Office of the Chief Engineer.

1.2.3 The NASA Headquarters' Chief Engineer shall periodically benchmark each Center's software engineering capability against its Center Software Engineering Improvement Plan. [SWE-004]

Note: Center Software Engineering Improvement Plans are documented per Center Software Engineering

Improvement Plan requirements. Capability Maturity Model® Integration (CMMI®) for Development (CMMI-DEV) appraisals are the preferred benchmarks for objectively measuring progress toward software engineering process improvement at NASA Centers.

1.2.4 Each Center shall establish, document, execute, and maintain software processes. [SWE-005]

1.2.5 To support compliance with NASA policy and facilitate the application of resources to mitigate risk, the NASA Headquarters' Chief Engineer, in coordination with the Chief, Safety and Mission Assurance, shall maintain a reliable list of the Agency's programs and projects containing software. [SWE-006]

1.3 Hierarchy of NASA Software-Related Documents

This paragraph helps the reader understand the flow down of requirements with respect to software created and acquired by or for NASA. Figure 1-1 shows the software engineering perspective of the relationship between relevant documents. The shaded documents in the figure show documents that primarily address software engineering policy and requirements. The text that follows the figure provides a brief description of each type of document, listed according to its position in the figure.

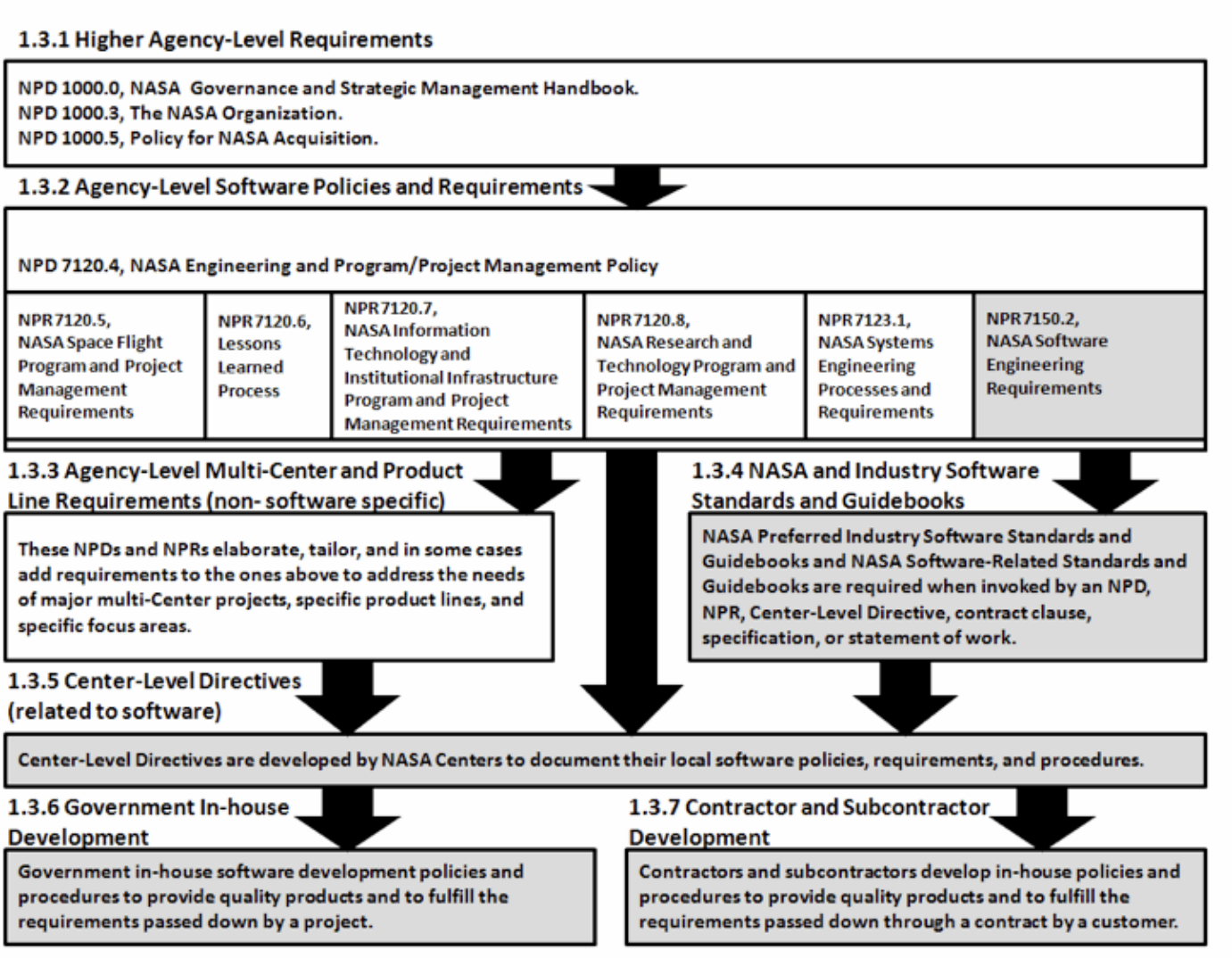

FIGURE 1-1 Relationships of Governing Software Documents

1.3.1 Higher Agency-Level Requirements

NPD 1000.0, Strategic Management and Governance Handbook, is the highest ranking NASA directive.

NPD 1000.0 sets forth the principles by which NASA will strategically manage the Agency, describes the means for doing so, and identifies the specific requirements that drive NASA's strategic planning process, leading to products such as the Strategic Plan and the Annual Performance and Accountability Report. NPD 1000.3 defines the basic roles and responsibilities necessary to conduct the mission and business of NASA. It is the official repository for defining NASA's organizational architecture. NPD 1000.5 provides the overall policy framework of NASA's disciplined, comprehensive strategic acquisition process with appropriate references to other key processes and directives. This acquisition process complies with NASA obligations as a Federal agency and is tailored to each of NASA's major areas of investment to ensure the efficient, effective use of the resources entrusted to the Agency. In the event of a conflict among the top-level directives, the information provided in the highest ranking directive takes precedence. In the event of conflict among the top-level directives and one or more lower-level NPDs and/or NPRs, the information provided in the top-level directive(s) takes precedence. These policies may include very high-level requirements relevant to software and information technology that are elaborated in lower-level policies and procedural requirements.

1.3.2 Agency-Level Software Policies and Requirements

NASA Policy Directive (NPD) 7120.4, NASA Engineering and Program/Project Management Policy, is an overarching document that establishes top-level policies for all software created and acquired by or for NASA. This policy covers software created, acquired, or maintained by or for NASA, including COTS, GOTS, and MOTS software, and open source, embedded, reused, and legacy/heritage software. NPR 7150.2, NASA Software Engineering Requirements, supports the implementation of the NPD 7120.4, NASA Engineering and Program/Project Management Policy. NPR 7150.2 establishes the NASA Software Classifications definitions and provides the minimal set of requirements established by the Agency for software acquisition, development, maintenance, retirement, operations, and management. NPR 7150.2 provides a set of software engineering requirements in generic terms to be applied throughout NASA and its contractor community. Additional Agency-level project management requirements (NPR 7120.5, NASA Space Flight Program and Project Management Requirements, NPR 7120.6, Lessons Learned Process, NPR 7120.7, NASA Information Technology and Institutional Infrastructure Program and Project Management Requirements, and NPR 7120.8, NASA Research and Technology Program and Project Management Requirements), information technology requirements (NPR 7120.7, NASA Information Technology and Institutional Infrastructure Program and Project Management Requirements) and system engineering requirements (NPR 7123, NASA Systems Engineering Processes and Requirements) exist that influence and affect the software development activities on a project. In the event of a conflict between an NPD and an NPR, the information provided in the NPD takes precedence.

1.3.3 Agency-Level Multi-Center and Product Line Requirements (non-software specific)

These NPDs and NPRs elaborate, tailor, and in some cases add requirements to the ones above to address the needs of major multi-Center projects, specific product lines, and specific focus areas. Examples of representative NPRs in this category are NPR 8705.2, Human-Rating Requirements for Space Systems, NPR 8715.3, NASA General Safety Program Requirements, and NPR 8735.2, Management of Government Quality Assurance Functions for NASA Contracts.

1.3.4 NASA and Industry Software Standards and Guidebooks

NASA Preferred Industry Software Standards and Guidebooks and NASA Software-Related Standards and Guidebooks are required when invoked by an NPD, NPR, Center-Level Directive, contract clause, specification, or statement of work.

1.3.5 Center-Level Directives (related to software)

Center-Level Directives are developed by NASA Centers to document their local software policies, requirements, and procedures. These directives are responsive to the requirements above them while addressing the specific application areas and the Center's mission within the Agency. In the event of a conflict between an NPD or an NPR with a Center-Level Directive, the information provided in the NPD or NPR takes precedence.

1.3.6 Government In-house Development

Government in-house software development policies and procedures are developed to provide quality software products that fulfill the requirements passed down by the project. Government in-house software development policies and procedures are typically designed to meet the needs of the supported projects in an effective and efficient manner.

1.3.7 Contractor and Subcontractor Development

Contractors and subcontractors develop in-house policies and procedures to provide quality software products and to fulfill the requirements passed down through a contract by a customer. Contractor and subcontractor policies and procedures are typically designed to satisfy different customers in an effective and efficient manner.

Chapter 2: Software Management Requirements

The software management activities define and control the many software aspects of a project from beginning to end. This includes the interfaces to other organizations, determination of deliverables, estimates and tracking of schedule and cost, risk management, formal and informal reviews as well as other forms of verification and validation, and determination of the amount of supporting services. The planned management of these activities is captured in one or more software and/or system plans.

2.1 Compliance with Laws, Policies, and Requirements

The software management process requires the understanding and application of laws and additional NASA policy requirements that impact the development, release, and/or maintenance of software. The documents listed in this section are additional requirements that may have an affect on software development projects and are mentioned here for awareness and completeness.

2.1.1 The project ensures that software invention requirements of NPD 2091.1, Inventions by Government Employees, are implemented by the project.

2.1.2 The project ensures that software technology transfer requirements of NPR 2190.1, NASA Export Control Program, are implemented by the project. The project ensures that there will be no access by foreign persons or export or transfer to foreign persons or destinations until an export control review is completed and access/release is approved in accordance with NPR 2190.1, NASA Export Control Program, and NPR 2210.1, External Release of Software.

2.1.3 The project ensures that software external release requirements of NPR 2210.1, External Release of NASA Software, are implemented by the project.

2.1.4 The project ensures that the information security requirements of NPD 2810.1, Security of Information Technology, are implemented by the project.

2.1.5 The project ensures that software is accessible to individuals with disabilities in accordance with 36 CFR Part 1194, Electronic and Information Technology Accessibility Standards.

2.1.6 The project ensures that software acquisitions or developments that meet NASA's capitalization criteria be capitalized per NASA's Property Plant and Equipment Policy, NPD 9250.1.

[Requirement numbers SWE-007 through SWE-012 are reserved.]

2.2 Software Life-Cycle Planning

Software Life-Cycle Planning covers the software aspects of a project from inception through retirement. Software Life-Cycle Planning cycle is an organizing process that considers the software as a whole and provides the planning activities required to insure a coordinated, well-engineered process for defining and implementing project activities. These processes, plans, and activities are coordinated within the project. At project conception, software needs for the project are analyzed, including acquisition, supply, development, operation, maintenance, retirement, and supporting activities and processes. The software effort is scoped and the processes, measurements, and

activities are documented in software plan(s).

2.2.1 Software Plans

2.2.1.1 The project shall develop software plan(s). [SWE-013]

Note: The requirement for the content of each software plan (whether stand-alone or condensed into one or more project level or software documents) is defined in Chapter 5. These include, but are not limited to:

a. Software development or management plan.

b. Software configuration management plan.

c. Software test plans.

d. Software maintenance plans.

e. Software assurance plans.

Note: Software engineering and the software assurance disciplines are integrally related and yet each has its own responsibilities. Jointly they are responsible for providing project management with the optimal solution for software to meet the engineering, safety, quality, and reliability needs of the project. This necessitates a close working relationship to assure the appropriate levels of effort for both. See NASA-STD-8739.8 for more details on development of a software assurance plan.

2.2.1.2 For safety-critical software, the project shall develop a software safety plan. [SWE-130]

Note: The requirement for the content of the software safety plan (whether stand-alone or condensed into one or more project level or software documents) is defined in Chapter 5. The NASA Software Safety Standard, NASA-STD-8719.13, contains detailed requirements and guidance on development of software safety plans. Software engineering and the software safety disciplines jointly are responsible for providing project management with the optimal solution for software to meet the engineering, safety, quality, and reliability needs of the project.

2.2.1.3 If a project is selected for software Independent Verification and Validation (IV&V) by the NASA Chief, Safety and Mission Assurance, the NASA IV&V program shall develop an IV&V Project Execution Plan (IPEP). [SWE-131]

Note: The selection of projects and capabilities by the Chief, Safety and Mission Assurance, for IV&V is based on recommendations from the NASA IV&V Board of Advisors. The recommendations provided by the NASA IV&V Board of Advisors are the result of a multi-step process. First, the NASA Software Inventory, maintained by the Office of the Chief Engineer, is used to generate a candidate list of projects. The candidate project criteria include:

a. Whether the project contains safety-critical software.

b. The project's software classification(s) per this NPR.

c. The level of software effort on the project.

d. The project's NASA Category (e.g., Category 1, 2, or 3, as defined in NPR 7120.5D).

e. Whether the responsible Center recommends the project for IV&V.

Second, the IV&V program utilizes this list and associated Center-provided data, captured in the NASA Software Inventory, to initiate a portfolio based risk assessment (PBRA) for capabilities within the candidate projects. The PBRA process assesses the risk of capabilities which have

software associated with them, based on factors including:

a. Complexity of the software implementing the capability under assessment.

b. Risk to safety and mission success associated with the capability under assessment.

c. Consequences of failure of the capability under assessment.

d. Time to criticality (i.e., the amount of time before the system or subsystem enters into a critical condition).

The results of these assessments are prioritized, by capability, in order to provide recommendations for IV&V support on specific capabilities associated with specific projects. These recommendations are provided to the NASA IV&V Board of Advisors, which in turn provides its recommendations to the Chief, Safety and Mission Assurance.

Note: The IV&V program determines and documents the services to be provided for projects selected for IV&V by the NASA Chief, Safety and Mission Assurance. IV&V support is funded and managed independent of the selected project. The IPEP is developed by the IV&V program and serves as the operational document that will be provided to the project receiving IV&V support. The IV&V program and the project will establish a mutual understanding of the NASA IV&V program's activities and IV&V project interfaces. Per the responsibilities defined in NPD 7120.4, NASA Engineering and Program/Project Management Policy, section 5.J.(5), projects ensure that software providers allow access to software and associated artifacts to enable implementation of IV&V. Additional information on the content of the IPEP is found in Chapter 5. Additional detail on the IV&V PBRA and IPEP may be found in the NASA IV&V Management System, located at: http://www.nasa.gov/centers/ivv/ims/home/index.html

2.2.2 The project shall implement, maintain, and execute the software plan(s). [SWE-014]

2.2.3 The project shall establish, document, and maintain at least one software cost estimate and associated cost parameter(s) that satisfies the following conditions: [SWE-015]

a. Covers the entire software life cycle.

b. Is based on selected project attributes (e.g., assessment of the size, functionality, complexity, criticality, and risk of the software processes and products).

c. Is based on the cost implications of the technology to be used and the required maturation of that technology.

Note: In the event there is a decision to outsource, it is a best practice that both the acquirer (NASA) and the provider (contractor/sub) should be responsible for developing software cost estimates.

2.2.4 The project shall document and maintain a software schedule that satisfies the following conditions: [SWE-016]

a. Coordinates with the overall project schedule.

b. Documents the interactions of milestones and deliverables between software, hardware, operations, and the rest of the system.

c. Reflects the critical path for the software development activities.

2.2.5 The project shall plan, track, and ensure project specific software training for project personnel. [SWE-017]

Note: This requirement is intended to address skills needed to support the software activities on a project. Not all skills required to accomplish a software project are "software" skills. The software engineering staff may require training in other domains to facilitate support of a project.

2.2.6 The project shall regularly hold reviews of software activities, status, and results with the project stakeholders and track issues to resolution. [SWE-018]

2.2.7 The project shall select and document a software development life cycle or model that includes phase transition criteria for each life-cycle phase (e.g., formal review milestones, informal reviews, software requirements review (SRR), preliminary design review (PDR), critical design review (CDR), test readiness reviews, customer acceptance or approval reviews). [SWE-019]

2.2.8 Software Assessments

2.2.8.1 The project shall classify each system and subsystem containing software in accordance with the software classification definitions for Classes A, B, C, D, E, F, G, and H software in Appendix E. [SWE-020]

Note: The applicability of requirements in this NPR to specific systems and subsystems containing software is determined through the use of the NASA-wide definitions for software classes in Appendix E and the designation of the software as safety-critical or non safety-critical in conjunction with the Requirements Mapping Matrix in Appendix D. These definitions are based on 1) usage of the software with or within a NASA system, 2) criticality of the system to NASA's major programs and projects, 3) extent to which humans depend upon the system, 4) developmental and operational complexity, and 5) extent of the Agency's investment. The NPR allows software written to support development activities (e.g., verification software, simulations) to be classified separately from the system under development; however, such support software is usually developed in conjunction with the system and not as a separate project. Projects may decide to reuse processes and work products established for the development of the system to satisfy the NPR 7150.2 requirements for the support software. The software assurance organization will perform an independent classification assessment and the results will be compared as per NASA-STD-8739.8, Software Assurance Standard. Software management and software assurance must reach agreement on classification of systems and subsystems. Disagreements are elevated via both the Engineering Technical Authority and Safety and Mission Assurance Technical Authority chains. The classification decision is documented in the Software Development or Management Plan as defined in Chapter 5.

2.2.8.2 The project's software assurance organization shall perform an independent classification assessment. [SWE-132]

2.2.8.3 The project, in conjunction with the Safety and Mission Assurance organization, shall determine the software safety criticality in accordance with NASA-STD-8739.8. [SWE-133]

Note: Software safety criticality is initially determined in the formulation phase using the NASA Software Assurance Standard, NASA-STD-8739.8. As the software is developed or changed and the computer software configuration items (CSCI), models, and simulations are identified, the safety-critical software determination can be reassessed and applied at lower levels. The software safety assessment and planning are performed for each software acquisition, development, and maintenance activity, and for changes to legacy\heritage systems. When software in a system or subsystem is found to be safety critical, additional requirements in the NASA Software Safety Standard will augment those associated with the software class requirements found in this document. The software assurance organization is required by NASA Software Assurance Standard, NASA-STD-8739.8, to perform an independent software safety criticality assessment and work with the project to resolve any differences. Engineering and software assurance must reach agreement on

safety-critical determination of software. Disagreements are elevated via both the Engineering Technical Authority and Safety and Mission Assurance Technical Authority chains.

2.2.9 If a system or subsystem evolves to a higher software classification as defined in Appendix E, then the project shall update its plan to fulfill the added requirements per the Requirements Mapping Matrix in Appendix D. [SWE-021]

2.2.10 The project shall implement software assurance per NASA-STD-8739.8, NASA Software Assurance Standard. [SWE-022]

Note: Software assurance activities occur throughout the life of the project. Some of the actual analyses and activities may be performed by engineering or the project. NASA's Safety and Mission Assurance organizations provide assurance that the products and processes are implemented according to the agreed upon plan(s). Software assurance is recommended on software activities and products, including solicitations, contract and memorandums of agreements, software plans, requirements, design, implementation, verification, validation, certification, acceptance, maintenance, operations, and retirement activities.

2.2.11 When a project is determined to have safety-critical software, the project shall ensure that the safety requirements of NASA-STD-8719.13, Software Safety Standard, are implemented by the project. [SWE-023]

NOTE: Engineering and software assurance initially determine software safety criticality in the formulation phase per NASA-STD-8739.8, NASA Software Assurance Standard; the results are compared and any differences are resolved. As the software is developed or changed and the software components, software models, and software simulations are identified, the safety-critical software determination can be reassessed and applied at lower levels. Further scoping and tailoring of the safety effort is found in the NASA-STD-8719.13, Software Safety Standard and NASA-GB-8719.13, NASA Software Safety Guidebook.

2.2.12 When a project is determined to have safety-critical software, the project shall ensure the following items are implemented in the software: [SWE-134]

a. Safety-critical software is initialized, at first start and at restarts, to a known safe state.

b. Safety-critical software safely transitions between all predefined known states.

c. Termination performed by software of safety critical functions is performed to a known safe state.

d. Operator overrides of safety-critical software functions require at least two independent actions by an operator.

e. Safety-critical software rejects commands received out of sequence, when execution of those commands out of sequence can cause a hazard.

f. Safety-critical software detects inadvertent memory modification and recovers to a known safe state.

g. Safety-critical software performs integrity checks on inputs and outputs to/from the software system.

h. Safety-critical software performs prerequisite checks prior to the execution of safety-critical software commands.

i. No single software event or action is allowed to initiate an identified hazard.

j. Safety-critical software responds to an off nominal condition within the time needed to prevent a

hazardous event.

k. Software provides error handling of safety-critical functions.

l. Safety-critical software has the capability to place the system into a safe state.

m. Safety-critical elements (requirements, design elements, code components, and interfaces) are uniquely identified as safety-critical. n. Incorporate requirements in the coding methods, standards, and/or criteria to clearly identify safety-critical code and data within source code comments.

Note: This section provides additional software safety requirements that are considered a best practice for safety-critical systems incorporating safety-critical software. These requirements are applicable to components that reside in a safety-critical system, and the components control, mitigate or contribute to a hazard as well as software used to command hazardous operations/activities. The requirements contained in this section complement the processes identified in NASA-STD-8719.13, NASA Software Safety Standard. Software engineering and software assurance disciplines each have specific responsibilities for providing project management with work products that meet the engineering, safety, quality, and reliability requirements on a project.

Note: Additional notes on specific items addressed above are:

Item a. - Aspects to consider when establishing a known safe state includes state of the hardware and software, operational phase, device capability, configuration, file allocation tables, and boot code in memory.

Item d. - Multiple independent actions by the operator help to reduce potential operator mistakes.

Item f. - Memory modifications may occur due to radiation-induced errors, uplink errors, configuration errors, or other causes so the computing system must be able to detect the problem and recover to a safe state. As an example, computing systems may implement error detection and correction, software executable and data load authentication, periodic memory scrub, and space partitioning to provide protection against inadvertent memory modification. Features of the processor and/or operating system can be utilized to protect against incorrect memory use.

Item g. - Software needs to accommodate both nominal inputs (within specifications) and off-nominal inputs, from which recovery may be required.

Item h. - The requirement is intended to preclude the inappropriate sequencing of commands. Appropriateness is determined by the project and conditions designed into the safety-critical system. Safety-critical software commands are commands that can cause or contribute to a hazardous event or operation.

Item j. - The intent is to establish a safe state following detection of an off-nominal indication. The safety mitigation must complete between the time that the off-nominal condition is detected and the time the hazard would occur without the mitigation. The safe state can either be an alternate state from normal operations or can be accomplished by detecting and correcting the fault or failure within the timeframe necessary to prevent a hazard and continuing with normal operations.

Item k.- Error handling is an implementation mechanism or design technique by which software faults and/or failures are detected, isolated, and recovered to allow for correct run-time program execution. The software error handling features that support safety-critical functions may detect and respond to hardware and operational faults and/or failures.

Item l.- The design of the system must provide sufficient sensors and effectors, as well as self checks within the software, in order to enable the software to detect and respond to system potential hazards.

2.2.13 The project shall ensure that actual results and performance of software activities are tracked against the software plans. [SWE-024]

2.2.14 The project shall ensure that corrective actions are taken, recorded, and managed to closure when actual results and performance deviate from the software plans. [SWE-025]

2.2.15 The project shall ensure that changes to commitments (e.g., software plans) are agreed to by the affected groups and individuals. [SWE-026]

2.3 Commercial, Government, Legacy\Heritage and Modified Off-The-Shelf Software

Projects utilizing Commercial, Government, Legacy\Heritage, and MOTS software components need to take into consideration the importance of planning and managing the inclusion of those components into the project software. The off-the-shelf software discussed here apply only when the off-the-shelf software elements are to be included as part of a NASA system (per section P.2.1). The following requirements do not apply to stand-alone desktop applications (e.g., word processing programs, spreadsheet programs, presentation programs). When software components use COTS applications (e.g., spreadsheet programs, database programs) within a NASA system/subsystem application, the software components need to be assessed and classified as part of the software subsystem in which they reside. Note that Commercial, Government, Legacy\Heritage, and MOTS software must also meet the applicable requirements for each class of software.

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.

Note: 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.

Note: 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. 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. Documentation changes 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. Open source software licenses review by the Center Counsel.

2.4 Software Verification and Validation

Ensuring that the software products meet their requirements and intended usage, and that the products were built correctly is the purpose of verification and validation. Both software validation and software verification activities span the entire software life cycle and need to be planned. Formal and informal reviews, software peer reviews/inspections, testing, demonstration, and analyses can be used. Each project is generally free to choose the extent and combination of verification and validation methods and activities that best suit the project. Because software peer reviews/inspections are such an important verification and validation tool with proven value, specific software peer review/inspection requirements are contained in this NPR (Chapter 4).

2.4.1 The project shall plan software verification activities, methods, environments, and criteria for the project. [SWE-028]

Note: Software verification is a software engineering activity that shows confirmation that software products properly reflect the requirements specified for them. In other words, verification ensures that "you built it right." Examples of verification methods include but are not limited to: software peer reviews/inspections of software engineering products for discovery of defects, software verification of requirements by use of simulations, black box and white box testing techniques, software load testing, software stress testing, software performance testing, decision table-based testing, functional decomposition-based testing, acceptance testing, path coverage testing, analyses of requirement implementation, and software product demonstrations. Refer to the software plan requirements for software verification planning and incorporation, including the planned use of software IV&V activities.

2.4.2 The project shall plan the software validation activities, methods, environments, and criteria for the project. [SWE-029]

Note: 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 (Chapter 5).

2.4.3 The project shall record, address, and track to closure the results of software verification activities. [SWE-030]

2.4.4 The project shall record, address, and track to closure the results of software validation activities. [SWE-031]

2.5 Project Formulation Requirements

Much of the project preparation and planning takes place during project formulation. Identification of project formulation requirements is an essential part of the early planning phases and must be started as the project begins. This is especially important as software requirements must be properly incorporated into the project cost estimates, schedule estimates, work planning, solicitations, evaluations of contractors, and the contracts themselves.

2.5.1 [SWE-032] The project shall ensure that software is acquired, developed, and maintained by an organization with a non-expired Capability Maturity Model Integration® for Development (CMMI-DEV) rating as measured by a Software Engineering Institute (SEI) authorized or certified lead appraiser as follows:

For **Class A** software:

CMMI-DEV Maturity Level 3 Rating or higher for software, or CMMI-DEV Capability Level 3 Rating or higher in all CMMI-DEV Maturity Level 2 and 3 process areas for software.

For **Class B** software:

CMMI-DEV Maturity Level 2 Rating or higher for software, or CMMI-DEV Capability Level 2 Rating or higher for software in the following process areas:

a. Requirements Management.

b. Configuration Management.

c. Process and Product Quality Assurance.

d. Measurement and Analysis.

e. Project Planning.

f. Project Monitoring and Control.

g. Supplier Agreement Management (if applicable).

For **Class C** software:

The required CMMI-DEV Maturity Level for Class C software will be defined per Center or project requirements.

Note: Organizations who have completed Standard CMMI® Appraisal Method for Process Improvement (SCAMPISM) Class A appraisals against the CMMI-DEV model are to maintain their rating and have their results posted on the SEI Web site so that NASA can assess the current maturity/capability rating. Software development organizations need to be reappraised and keep an active appraisal rating posted on the SEI Web site during the time that they are responsible for the development and maintenance of the software.

Note: For Class A software development only, a transition period to obtain a CMMI-DEV

Maturity/Capability Level 3 Rating will be allowed for organizations developing Class A software per the NASA Headquarters' Office of the Chief Engineer's approved Center Software Engineering Improvement Plan as described in SWE-003, SWE-004, and SWE-108.

Note: For Class B software, in lieu of a CMMI rating by a development organization, the project will conduct an evaluation, performed by a qualified evaluator selected by the Center Engineering Technical Authority, of the seven process areas listed in SWE-032 and mitigate any risk, if deficient. This exception is intended to be used in those cases in which NASA wishes to purchase a product from the "best of class provider," but the best of class provider does not have the required CMMI rating. When this exception is exercised, the Center Engineering Technical Authority should be notified.

2.5.2 The project shall assess options for software acquisition versus development. [SWE-033]

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

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

b. Develop the software product or obtain the software service internally.

c. Develop the software product or obtain the software service through contract.

d. Enhance an existing software product or service.

Note: 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.

2.5.3 The project shall define and document or record the acceptance criteria and conditions for the software. [SWE-034]

2.5.4 For new contracts, the project shall establish a procedure for software supplier selection, including proposal evaluation criteria. [SWE-035]

2.5.5 The project shall determine which software processes, activities, and tasks are required for the project. [SWE-036]

2.5.6 The project shall define the milestones at which the software supplier(s) progress will be reviewed and audited as a part of the acquisition activities. [SWE-037]

Note: Known contract milestones are expected to be included in the resulting contract.

2.5.7 The project shall document software acquisition planning decisions. [SWE-038]

Note: This may be in an acquisition plan or in another project planning document.

2.6 Software Contract Requirements

The requirements in this section are applicable for NASA contracted software procurements (e.g., reuse of existing software, modification of existing software, contracted and subcontracted software, and/or development of new software). Acquisition requirements are focused both inside the acquisition organization to ensure the acquisition is conducted effectively and outside the acquisition organization as the organization conducts project monitoring and control of its suppliers. These

acquisition requirements provide a foundation for acquisition process discipline and rigor that enables product and service development to be repeatedly executed with high levels of acquisition success. This section contains project software acquisition and contract requirements to ensure that NASA has the data needed for the review of project provided systems and/or services. The project is responsible for ensuring that these requirements apply when software activities are subcontracted from a NASA prime contractor. These requirements are used in addition to, not in place of, the other requirements of this NPR.

2.6.1 Government software insight requirements.

2.6.1.1 The project shall require the software supplier(s) to provide insight into software development and test activities; at a minimum the following activities are required: monitoring integration, review of the verification adequacy, review of trade study data and results, auditing the software development process, participation in software reviews and systems and software technical interchange meetings. [SWE-039]

2.6.1.2 The project shall require the software supplier(s) to provide NASA with all software products and software process tracking information, in electronic format, including software development and management metrics. [SWE-040]

2.6.1.3 The project shall require the software supplier(s) to notify the project, in the response to the solicitation, as to whether open source software will be included in code developed for the project. [SWE-041]

2.6.1.4 The project shall require the software supplier(s) to provide NASA with electronic access to the source code developed for the project, including MOTS software and non-flight software (e.g., ground test software, simulations, ground analysis software, ground control software, science data processing software, and hardware manufacturing software). [SWE-042]

Note: Known contract requirements are addressed in the solicitations and included in the resulting contract. Additionally, if the project needs to control further use and distribution of the resulting software or requires unlimited rights in the software, e.g., right to use, modify, and distribute the software for any purpose, the project can consider having the software copyright assigned to the Government. A list of software deliverables needs to be addressed in the solicitations, if the software is being procured. The project can consult with the Center's Chief of Patent/Intellectual Property Counsel regarding required rights associated with the software. Section 3.5.4 contains a list of Software Operations, Maintenance, and Retirement deliverables.

2.6.2 Supplier Monitoring Requirements.

2.6.2.1 The project shall require the software supplier to track all software changes and non-conformances and provide the data for the project's review. [SWE-043]

2.6.2.2 The project shall require the software supplier(s) to provide software metric data as defined in the project's Software Metrics Report. [SWE-044]

Note: The requirement for the content of a Software Metrics Report is defined in Chapter 5.

2.6.2.3 The project shall participate in any joint NASA/contractor audits of the software development process and software configuration management process. [SWE-045]

2.6.2.4 The project shall require the software supplier(s) to provide a software schedule for the project's review and schedule updates as requested. [SWE-046]

2.6.2.5 The project shall require the software supplier(s) to make available, electronically, the software traceability data for the project's review. [SWE-047]

2.6.2.6 The project shall document in the solicitation the software processes, activities, and tasks to be performed by the supplier. [SWE-048]

Chapter 3: Software Engineering Life-Cycle Requirements

This NPR makes no recommendation for a specific software life-cycle model. Each has its strengths and weaknesses, and no one model is best for every situation. Whether using the spiral model, the iterative model, waterfall, or any other development life-cycle model, each has steps of requirements, design, implementation, testing, release to operations, maintenance, and retirement. Although this NPR does not impose a particular life-cycle model on each software project, this NPR does support a standard set of life-cycle phases. Use of the different phases of a life cycle allows the various products of a project to be gradually developed and matured from initial concepts through the fielding of the product and to its final retirement. Without recommending a life cycle, the requirements for each of these steps are provided below.

3.1 Software Requirements

The requirements phase is one of the most important phases of software engineering. Studies show that the top problems in the software industry are due to poor requirements elicitation, inadequate requirements specification, and inadequate management of changes to requirements. Requirements provide the foundation for the entire life cycle as well as for the software product. Requirements also provide a basis for planning and estimating. Requirements are based on customer, user, and other stakeholder needs and design and development constraints. The development of requirements includes elicitation, analysis, documentation, verification, and validation. Ongoing customer validation of the requirements to ensure the end products meet the customer needs is an important part of the life-cycle process. This can be accomplished via rapid prototyping and customer-involved reviews of iterative and final software requirements.

3.1.1 Requirements Development.

3.1.1.1 The project shall document the software requirements. [SWE-049]

Note: The requirements for the content of a Software Requirement Specification and a Data Dictionary document are defined in Chapter 5. The requirements definition activity also includes documenting key decisions, developing requirement rationales, and defining assumptions. The requirement development activities can use lessons learned in performing the logical decomposition process activities. The requirements definition activity provides an understanding of the derived technical requirements baseline, a logical decomposition model, traceability to technical requirements, and an understanding of the stakeholder's expectations.

3.1.1.2 The project shall identify, develop, document, approve, and maintain software requirements based on analysis of customer and other stakeholder requirements and the operational concepts. [SWE-050]

3.1.1.3 The project shall perform software requirements analysis based on flowed-down and derived requirements from the top-level systems engineering requirements and the hardware specifications and design. [SWE-051]

Note: The software requirements analysis determines the requirement's safety criticality, correctness, consistency, clarity, completeness, traceability, feasibility, verifiability, and maintainability. The software requirements analysis activities include the allocation of functional, non-functional, and performance requirements to functions and subfunctions.

3.1.1.4 The project shall perform, document, and maintain bidirectional traceability between the software requirement and the higher-level requirement. [SWE-052]

3.1.2 Requirements Management.

3.1.2.1 The project shall collect and manage changes to the software requirements. [SWE-053]

Note: The project analyzes and documents changes to requirements for cost, technical, and schedule impacts.

3.1.2.2 The project shall identify, initiate corrective actions, and track until closure inconsistencies among requirements, project plans, and software products. [SWE-054]

3.1.2.3 The project shall perform requirements validation to ensure that the software will perform as intended in the customer environment. [SWE-055]

Note: Requirements validation includes confirmation that the requirements meet the needs and expectations of the customer. Requirement validation is confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled.

3.2 Software Design

Software design is the process of defining the software architecture, components, modules, interfaces, and data for a software system to satisfy specified requirements. The software architecture is the fundamental organization of a system embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution. The software architectural design is concerned with creating a strong overall structure for software entities that fulfill allocated system and software-level requirements. Typical views captured in an architectural design include the decomposition of the software subsystem into design entities, computer software configuration items (CSCI), definitions of external and internal interfaces, dependency relationships among entities and system resources, and finite state machines. Detailed design further refines the design into lower-level entities that permit the implementation by coding in a programming language. Typical attributes that are documented for lower-level entities include: identifier, type, purpose, function, constraints, subordinates, dependencies, interface, resources, processing, and data. Rigorous specification languages, graphical representations, and related tools have been developed to support the evaluation of critical properties at the design level. Projects are encouraged to take advantage of these improved design techniques to prevent and eliminate errors as early in the life cycle as possible.

3.2.1 The project shall document and maintain the software design. [SWE-056]

Note: The requirement for the content of a Software Design Description document and an interface design description document are defined in Chapter 5 as applicable by software classification.

3.2.2 The project shall transform the allocated and derived requirements into a documented software architectural design. [SWE-057]

Note: The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the properties of those components, and the relationships between them. Documenting software architecture facilitates communication between stakeholders, documents early decisions about high-level design, and allows reuse of design components and patterns between projects.

3.2.3 The project shall develop, record, and maintain a detailed design based on the software architectural design that describes the lower-level units so that they can be coded, compiled, and tested. [SWE-058]

3.2.4 The project shall perform and maintain bidirectional traceability between the software requirements and the software design. [SWE-059]

3.3 Software Implementation

Software implementation consists of implementing the requirements and design into code, data, and documentation. Software implementation also consists of following coding methods and standards. Unit testing is also usually a part of software implementation (unit testing can also be conducted during the testing phase).

3.3.1 The project shall implement the software design into software code. [SWE-060]

3.3.2 The project shall ensure that software coding methods, standards, and/or criteria are adhered to and verified. [SWE-061]

3.3.3 The project shall ensure that results from static analysis tool(s) are used in verifying and validating software code. [SWE-135]

Note: Modern static code analysis tools can identify a variety of issues and problems, including but not limited to dead code, non-compliances with coding standards, security vulnerabilities, race conditions, memory leaks, and redundant code. Typically, static analysis tools are used to help verify adherence with coding methods, standards, and/or criteria. While false positives are an acknowledged shortcoming of static analysis tools, users can calibrate, tune, and filter results to make effective use of these tools. Software peer reviews/inspections of code items can include reviewing the results from static code analysis tools.

Note: Static analysis tools may not be readily available for some platforms or computing languages. If static analysis tools are determined to not be available, the project can document the alternate manual methods and procedures to be used to verify and validate the software code. These manual methods and procedures will be addressed or referenced in the project's compliance matrix against this requirement.

3.3.4 The project shall ensure that the software code is unit tested per the plans for software testing. [SWE-062]

3.3.5 The project shall provide a Software Version Description document for each software release. [SWE-063]

Note: The requirement for the content of a Software Version Description document is defined in Chapter 5.

3.3.6 The project shall provide and maintain bidirectional traceability from software design to the software code. [SWE-064]

3.3.7 The project shall validate and accredit software tool(s) required to develop or maintain software. [SWE-136]

Note: Projects need to determine and define acceptance processes for software tool(s), used to develop or maintain Class A, B, C, or safety-critical software. Examples of software tools include but are not limited to compilers, code-coverage tools, development environments, build tools, user interface tools, debuggers, and code generation tools.

3.4 Software Testing

The purpose of testing is to verify the software functionality and remove defects. Testing verifies the code against the requirements and the design to ensure that the requirements are implemented. Testing also identifies problems and defects that are corrected and tracked to closure before product delivery. Testing also validates that the software operates appropriately in the intended environment.

3.4.1 The project shall establish and maintain: [SWE-065]

a. Software Test Plan(s).

b. Software Test Procedure(s).

c. Software Test Report(s).

Note: The requirements for the content of a Software Test Plan, Software Test Procedure, and Software Test Report are defined in Chapter 5.

3.4.2 The project shall perform software testing as defined in the Software Test Plan. [SWE-066]

Note: A best practice for Class A, B, and C software projects is to have formal software testing conducted, witnessed, and approved by an independent organization outside of the development team. Testing could include software integration testing, systems integration testing, validation testing, end-to-end testing, acceptance testing, white and black box testing, decision and path analysis, statistical testing, stress testing, performance testing, regression testing, qualification testing, simulation, and others. Use of automated software testing tools are also to be considered in software testing. Test breadth and accuracy can be increased through the use of test personnel independent of the software design and implementation teams, software peer reviews/inspections of Software Test Procedures and Software Test Results, and employing impartial test witnesses.

3.4.3 The project shall ensure that the implementation of each software requirement is verified to the requirement. [SWE-067]

3.4.4 The project shall evaluate test results and document the evaluation. [SWE-068]

3.4.5 The project shall document defects identified during testing and track to closure. [SWE-069]

3.4.6 The project shall verify, validate, and accredit software models, simulations, and analysis tools required to perform qualification of flight software or flight equipment. [SWE-070]

Note: Center processes address issues such as numerical accuracy, uncertainty analysis, and sensitivity analysis, as well as verification and validation for software implementations of models and simulations. Information regarding specific verification and validation techniques and the analysis of models and simulations can be found in the NASA standard NASA-STD-7009.

3.4.7 The project shall update Software Test Plan(s) and Software Test Procedure(s) to be consistent with software requirements. [SWE-071]

3.4.8 The project shall provide and maintain bidirectional traceability from the Software Test Procedures to the software requirements. [SWE-072]

3.4.9 The project shall ensure that the software system is validated on the targeted platform or high-fidelity simulation. [SWE-073]

Note: Typically, a high-fidelity simulation has the exact processor, processor performance, timing, memory size, and interfaces as the flight unit.

3.5 Software Operations, Maintenance, and Retirement

Planning for operations, maintenance, and retirement must be considered throughout the software life cycle. Operational concepts and scenarios are derived from customer requirements and validated in the operational or simulated environment. Software maintenance activities sustain the software product after the product is delivered to the customer until retirement.

3.5.1 The project shall document the software maintenance plans in a Software Maintenance Plan document. [SWE-074]

Note: The requirement for the content of a Software Maintenance Plan is defined in Chapter 5.

3.5.2 The project shall plan software operations, maintenance, and retirement activities. [SWE-075]

3.5.3 The project shall implement software operations, maintenance, and retirement activities as defined in the respective plans. [SWE-076]

3.5.4 The project shall complete and deliver the software product to the customer with appropriate documentation to support the operations and maintenance phase of the software's life cycle. [SWE-077]

Note: Delivery includes, as applicable, Software User's Manual (as defined in Chapter 5), source files, executable software, procedures for creating executable software, procedures for modifying the software, and a Software Version Description. Open source software licenses are reviewed by the Center's Chief of Patent/Intellectual Property Counsel before being accepted into software development projects. Other documentation considered for delivery includes:

a. Summary and status of all accepted Change Requests to the baselined Software Requirements Specifications.

b. Summary and status of all major software capability changes since baselining of the Software Design Documents.

c. Summary and status of all major software tests (including development, verification, and performance testing).

d. Summary and status of all Problem Reports written against the software.

e. Summary and status of all software requirements deviations and waivers.

f. Summary and status of all software user notes.

g. Summary and status of all quality measures historically and for this software.

h. Definition of open work, if any.

i. Software configuration records defining the verified and validated software, including requirements verification data (e.g., requirements verification matrix).

j. Final version of the software documentation, including the final Software Version Description document(s).

k. Summary and status of any open software-related risks.

3.5.5 The project shall deliver to the customer the as-built documentation to support the operations and maintenance phase of the software life cycle. [SWE-078]

Chapter 4: Supporting Software Life-Cycle Requirements

Support processes are not limited to a single software life-cycle phase such as requirements, design, implementation, or test. Support processes typically occur throughout the software life cycle. For example, typical configuration management baselines (e.g., requirements, code, products) happen across the life cycle. Support processes are software management and engineering processes that typically support the entire software life cycle (e.g., configuration management).

4.1 Software Configuration Management

Software configuration management is the process of applying configuration management throughout the software life cycle to ensure the completeness and correctness of software configuration items. Software configuration management applies technical and administrative direction and surveillance to: identify and document the functional and physical characteristics of software configuration items, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements. Software configuration management establishes and maintains the integrity of the products of a software project throughout the software life cycle. Use of standard Center or organizational software configuration management processes and procedures is encouraged where applicable.

4.1.1 The project shall develop a Software Configuration Management Plan that describes the functions, responsibilities, and authority for the implementation of software configuration management for the project. [SWE-079]

Note: The Software Configuration Management Plan may be a part of the project configuration management plan. The content is defined by the requirement in Chapter 5.

4.1.2 The project shall track and evaluate changes to software products. [SWE-080]

Note: The project can use a software change request system or a software problem tracking system. The minimum content for software change requests or a software problem report is defined in Chapter 5.

4.1.3 The project shall identify the software configuration items (e.g., software documents, code, data, tools, models, scripts) and their versions to be controlled for the project. [SWE-081]

Note: The project is responsible for assuring that software safety elements are properly identified and controlled.

4.1.4 The project shall establish and implement procedures designating the levels of control each identified configuration item must pass through; the persons or groups with authority to authorize changes and to make changes at each level; and the steps to be followed to request authorization for changes, process change requests, track changes, distribute changes, and maintain past versions. [SWE-082]

4.1.5 The project shall prepare and maintain records of the configuration status of configuration items. [SWE-083]

Note: Configuration status accounting generates and/or maintains records of the status and contents of the software throughout the life cycle. This function keeps track of the changes and the contents

of versions and releases.

4.1.6 The project shall ensure that software configuration audits are performed to determine the correct version of the configuration items and verify that they conform to the documents that define them. [SWE-084]

4.1.7 The project shall establish and implement procedures for the storage, handling, delivery, release, and maintenance of deliverable software products. [SWE-085]

4.2 Risk Management

Identification and management of risks provide a basis for systematically examining changing situations over time to uncover and correct circumstances that impact the ability of the project to meet its objectives.

4.2.1 The project shall identify, analyze, plan, track, control, communicate, and document software risks in accordance with NPR 8000.4, Agency Risk Management Procedural Requirements. [SWE-086]

Note: A project needs to include an assessment of the risk that any untested code poses to the system or subsystem.

4.3 Software Peer Reviews/Inspections

Software peer reviews and inspections are the in-process technical examination of work products by peers to find and eliminate defects early in the life cycle. Software peer reviews/inspections are performed following defined procedures covering the preparation for the review, conducting the review itself, documenting results, reporting the results, and certifying the completion criteria. When planning the composition of a software peer review/inspection team consider including software testing, system testing, software assurance, software safety, and software Independent Verification and Validation (IV&V) personnel.

4.3.1 The project shall perform and report on software peer reviews/inspections for: [SWE-087]

a. Software requirements.

b. Software Test Plan.

c. Any design items that the project identified for software peer review/inspections according to the software development plans.

d. Software code as defined in the software and or project plans.

Note: Software peer reviews/inspections are a recommended best practice for all safety and mission-success related design and code software components. Guidelines for software peer reviews/inspections are contained in NASA-STD-2202-93, NASA Software Formal Inspection Standard.

4.3.2 The project shall perform and report on software peer reviews/inspections for: [SWE-137]

a. Software Development or Management Plan.

b. Software Configuration Management Plan.

c. Software Maintenance Plan.

d. Software Assurance Plan.

e. Software Safety Plan.

4.3.3 The project shall, for each planned software peer review/inspections: [SWE-088]

a. Use a checklist to evaluate the work products.

b. Use established readiness and completion criteria.

c. Track actions identified in the reviews until they are resolved.

d. Identify required participants.

4.3.4 The project shall, for each planned software peer review/inspection, record basic measurements. [SWE-089]

Note: The requirement for the content of a Software Peer Review/Inspection Report is defined in Chapter 5.

4.4 Software Measurement

Software measurement programs at multiple levels are established to meet measurement objectives. The requirements below are designed to establish measurement programs at the project and the Mission Directorate levels to assist in managing projects, assuring quality, and improving software engineering practices. Project-level and Mission Directorate/Mission Support Office-level (product line) measurement programs are designed to meet the following high-level goals:

a. To improve future planning and cost estimation.

b. To provide realistic data for progress tracking.

c. To provide indicators of software quality.

d. To provide baseline information for future process improvement activities.

Additional measures can be defined by either the projects or the Mission Directorate/Mission Support Office, based on any additional high-level goals they may have.

4.4.1 The project shall establish and document specific measurement objectives for their project. [SWE-090]

4.4.2 The project shall select and record the selection of specific measures in the following areas: [SWE-091]

a. Software progress tracking.

b. Software functionality.

c. Software quality.

d. Software requirements volatility.

e. Software characteristics.

Note: The requirement for a Software Metrics Report is defined in Chapter 5.

4.4.3 The project shall specify and record data collection and storage procedures for their selected

software measures and collect and store measures accordingly. [SWE-092]

4.4.4 The project shall analyze software measurement data collected using documented project-specified and Center/organizational analysis procedures. [SWE-093]

4.4.5 The project shall report measurement analysis results periodically and allow access to measurement information by Center-defined organizational measurement programs. [SWE-094]

4.4.6 Each NASA Mission Directorate shall establish its own software measurement system to include the minimum reporting requirements in SWE-091. [SWE-095]

4.4.7 Each NASA Mission Directorate shall identify and document the specific measurement objectives, the chosen specific measures, the collection procedures, and storage and analysis procedures. [SWE-096]

[Requirement number SWE-097 is reserved]

4.5 Best Practices

Best practices that are collected by the software community are a resource for improving software products. Ensuring an awareness of these practices can often provide potential solutions to problems. Successful best practices also provide alternate approaches for an individual project to consider, given its scope, domain, and goals. The intent of organizational best practices is not to mandate the use of any specific practice, but to provide information and examples to each project so that it can evaluate and choose those practices that it deems most beneficial. The identified best practices can then be used to improve the life-cycle processes for future software product improvement purposes.

4.5.1 The NASA Headquarters' Office of the Chief Engineer shall maintain an Agency-wide process asset library of applicable best practices. [SWE-098]

Note: The Agency-level process assets can be viewed from the NASA Software Process Asset Library (PAL) Web site, http://swpal.nasa.gov/, from the NASA Software Engineering Web site at http://software.nasa.gov and from the NASA Engineering Network Web site at http://nen.nasa.gov/portal/site/llis/OCE/. The repository may contain information in many forms including, but not limited to, processes, Web sites, design principles, books, periodicals, presentations, tools, examples of documents, and conference descriptions.

4.5.2 Each Center shall review the contents of the process asset library to identify those practices that may have direct applicability and value to its software activities. [SWE-099]

4.6 Training

Having properly trained personnel is one of the key items that can lead to success of software engineering projects. The goal is to maintain and advance organizational capability for training of personnel that perform software engineering practices to effectively meet scientific and technological objectives. The Software Training Plan includes training in the following software activities: software management, software acquisition, software monitoring, software development, software safety and mission assurance, and software process improvement.

4.6.1 The NASA Headquarters' Office of the Chief Engineer and Center training organizations shall provide and fund training to advance software engineering practices and software acquisition. [SWE-100]

4.6.2 Each Center shall maintain and implement Software Training Plan(s) to advance its in-house software engineering capability and as a reference for its contractors. [SWE-101]

Note: The Software Training Plan content is defined by the Software Training Plan requirement in Chapter 5.

Chapter 5: Software Documentation Requirements

Use of NASA Center and contractor formats in document deliverables will be acceptable if required content is addressed. Documents can be combined if required content is addressed. Specific content within these documents may not be applicable for every project. Non-applicable content areas will be clearly noted in project documentation. These non-applicable documentation decisions may be reviewed by external organizations. Product documentation are reviewed and updated as necessary.

5.1 Software Plans

5.1.1 Software Development or Management Plan.

The Software Development or Management Plan provides insight into, and a tool for monitoring, the processes to be followed for software development, the methods to be used, the approach to be followed for each activity, and project schedules, organization, and resources. This plan details the system software, project documentation, project schedules, resources requirements and constraints, and general and detailed software development activities.

5.1.1.1 The Software Development or Management Plan shall contain: [SWE-102]

a. Project organizational structure showing authority and responsibility of each organizational unit, including external organizations (e.g., Safety and Mission Assurance, Independent Verification and Validation (IV&V), Technical Authority, NASA Engineering and Safety Center, NASA Safety Center).

b. The safety criticality and classification of each of the systems and subsystems containing software.

c. Tailoring compliance matrix for approval by the designated Engineering Technical Authority, if the project has any waivers or deviations to this NPR.

d. Engineering environment (for development, operation, or maintenance, as applicable), including test environment, library, equipment, facilities, standards, procedures, and tools.

e. Work breakdown structure of the life-cycle processes and activities, including the software products, software services, non-deliverable items to be performed, budgets, staffing, acquisition approach, physical resources, software size, and schedules associated with the tasks.

f. Management of the quality characteristics of the software products or services.

g. Management of safety, security, privacy, and other critical requirements of the software products or services.

h. Subcontractor management, including subcontractor selection and involvement between the subcontractor and the acquirer, if any.

i. Verification and validation.

j. Acquirer involvement.

k. User involvement.

l. Risk management.

m. Security policy.

n. Approval required by such means as regulations, required certifications, proprietary, usage, ownership, warranty, and licensing rights.

o. Process for scheduling, tracking, and reporting.

p. Training of personnel, including project unique software training needs.

q. Software life-cycle model, including description of software integration and hardware/software integration processes, software delivery, and maintenance.

r. Configuration management.

s. Software documentation tree.

t. Software peer review/inspection process of software work products.

u. Process for early identification of testing requirements that drive software design decisions (e.g., special system level timing requirements/checkpoint restart).

v. Software metrics.

w. Content of software documentation to be developed on the project.

x. Management, development, and testing approach for handling any commercial-off-the-shelf (COTS), government-off-the-shelf (GOTS), modified-off-the-shelf (MOTS), reused, or open source software component(s) that are included within a NASA system or subsystem.

Note: Verification includes:

a. Identification of selected software verification methods and criteria across the life cycle (e.g., software peer review/inspections procedures, re-review/inspection criteria, testing procedures).

b. Identification of selected work products to be verified.

c. Description of software verification environments that are to be established for the project (e.g., software testing environment, system testing environment, regression testing environment).

d. Identification of where actual software verification records and analysis of the results will be documented (e.g., test records, software peer review/inspection records) and where software verification corrective action will be documented.

Note: Validation includes:

a. Identification of selected software validation methods and criteria across the life cycle (e.g., prototyping, user groups, simulation, analysis, acceptance testing, operational demonstrations).

b. Identification of selected work products to be validated.

c. Description of software validation environments that are to be established for the project (e.g., simulators for operational environment).

d. Identification of where actual software validation records and analysis of the results will be documented (e.g., user group records, prototyping records, and acceptance testing records) and where software validation corrective action will be documented.

5.1.2 Software Configuration Management Plan.

The Software Configuration Management Plan describes the functions, responsibilities, and authority for the accomplishment and implementation of software configuration management to be performed during the software life cycle. This plan identifies the required coordination of software configuration management activities with other activities of the project.

5.1.2.1 The Software Configuration Management Plan shall contain: [SWE-103]

a. The project organization(s).

b. Responsibilities of the software configuration management organization.

c. References to the software configuration management policies and directives that apply to the project.

d. All functions and tasks required to manage the configuration of the software, including configuration identification, configuration control, status accounting, and configuration audits and reviews.

e. Schedule information, which establishes the sequence and coordination for the identified activities and for all events affecting the plan's implementation.

f. Resource information, which identifies the software tools, techniques, and equipment necessary for the implementation of the activities.

g. Plan maintenance information, which identifies the activities and responsibilities necessary to ensure continued planning during the life cycle of the project.

h. Release management and delivery.

5.1.3 Software Test Plan

The Software Test Plan describes the plans for software component level testing, software integration testing, software qualification testing, and system qualification testing of software systems. The plan describes the software test environment to be used for testing, identifies the tests to be performed, and provides schedules for environment, development, and test activities. The plan provides an overview of software testing, test schedules, and test management procedures.

5.1.3.1 The Software Test Plan shall include: [SWE-104]

a. Test levels (separate test effort that has its own documentation and resources, e.g., component, integration, and system testing).

b. Test types:

(1) Unit testing.

(2) Software integration testing.

(3) Systems integration testing.

(4) End-to-end testing.

(5) Acceptance testing.

(6) Regression testing.

c. Test classes (designated grouping of test cases).

d. General test conditions.

e. Test progression.

f. Data recording, reduction, and analysis.

g. Test coverage (breadth and depth) or other methods for ensuring sufficiency of testing.

h. Planned tests, including items and their identifiers.

i. Test schedules.

j. Requirements traceability (or verification matrix).

k. Qualification testing environment, site, personnel, and participating organizations.

5.1.4 Software Maintenance Plan.

The Software Maintenance Plan provides insight into the method, approach, responsibility, and processes to be followed for maintenance of software and its associated documentation. For the Software Maintenance Plan, provide separate volumes for each system element (e.g., ground operations, flight operations, mission operations, and spacecraft).

5.1.4.1 The Software Maintenance Plan shall include: [SWE-105]

a. Plan information for the following activities:

(1) Maintenance process implementation.

(2) Problem and modification analysis.

(3) Modification implementation.

(4) Maintenance review/acceptance.

(5) Migration.

(6) Software Retirement.

(7) Software Assurance.

(8) Software Risk Assessment for all changes made during maintenance and operations.

b. Specific standards, methods, tools, actions, procedures, and responsibilities associated with the maintenance process. In addition, the following elements are included:

(1) Development and tracking of required upgrade intervals, including implementation plan.

(2) Approach for the scheduling, implementation, and tracking of software upgrades.

(3) Equipment and laboratories required for software verification and implementation.

(4) Updates to documentation for modified software components.

(5) Licensing agreements for software components.

(6) Plan for and tracking of operational backup software (e.g., backup flight software, backup to the primary operational software).

(7) Approach for the implementation of modifications to operational software (e.g., testing of

software in development laboratory prior to operational use).

(8) Approach for software delivery process, including distribution to facilities and users of the software products and installation of the software in the target environment (including, but not limited to, spacecraft, simulators, Mission Control Center, and ground operations facilities).

(9) Approach for providing NASA access to the software version description data (e.g., revision number, licensing agreement).

5.1.5 Software Assurance Plan.

The Software Assurance Plan details the procedures, reviews, and audits required to accomplish software assurance. The project office should coordinate with the Office of Safety and Mission Assurance for help in scoping and adapting the effort appropriately, and to designate the individual responsible for software assurance on the project.

5.1.5.1 The Software Assurance Plan(s) shall be developed and documented per NASA-STD-8739.8, NASA Software Assurance Standard. [SWE-106]

5.1.6 Center Software Training Plan.

5.1.6.1 The Center Software Training Plan shall include: [SWE-107]

a. Responsibilities.

b. Implementation.

c. Records and forms.

d. Training resources.

e. Minimum training requirements for software personnel.

f. Training class availabilities.

5.1.7 Center Software Engineering Improvement Plans

5.1.7.1 The Center Software Engineering Improvement Plans shall include: [SWE-108]

a. Process improvement goal(s).

b. Scope of process improvement.

c. All Center organizations responsible for the performance of mission-critical software development, management, and acquisition.

d. The Center's tactic for phasing in improvements (e.g., domain phasing and organizational phasing).

e. Ownership of Center Software Engineering Improvement Plan.

f. The Center's tactic for monitoring Center Software Engineering Improvement Plan progress, including responsibilities.

g. Strategies and objectives.

h. The Center's tactic for supporting the implementation of all strategies of the NASA Software Engineering Initiative Implementation Plan.

i. Schedule.

j. The Center's tactic or approach for phasing in new and upgraded NASA Headquarters requirements.

5.1.8 IV&V Project Execution Plan (IPEP).

The IPEP is divided into two major parts: the body of the document and the appendices. The body includes mission overview, IV&V goals and objectives, high-level description of the IV&V approach, associated milestones, activities and deliverables, resources, roles and responsibilities, and lines of communication to establish a formal mutual agreement, as necessary to execute the project. The appendices include information that will change over the course of the effort and include schedules, relevant IV&V Portfolio Based Risk Assessment (PBRA) results, IV&V Coverage Diagram Data, and an acronym list. Additional information on the IV&V PBRA and IPEP may be found in the NASA IV&V Management System, located at: http://www.nasa.gov/centers/ivv/ims/home/index.html.

5.1.9 Software Safety Plan.

This document details the activities, general relative schedule of needed activities, communication paths, and responsibilities for performing software safety activities as part of the systems safety program. This does not have to be a stand-alone document, but could be included as part of the systems safety plan or, for small projects, an overall assurance plan. While it may be developed either by the program/project/facility office or by the safety personnel within the Center Safety and Mission Assurance (SMA) organization(s), both the project or facility office and the Center SMA organization must concur.

5.1.9.1 The Software Safety Plan(s) shall be developed per NASA-STD-8719.13, Software Safety Standard. [SWE-138]

5.2 Software Requirements and Product Data

5.2.1 Software Requirements Specification.

The Software Requirements Specification details the software performance, interface, and operational and quality assurance requirements for each computer software configuration items (CSCI).

Note: Software requirements and design specifications need not be textual and may include representations in rigorous specification languages, graphical representations, or specifications suitable for requirements or design analysis tools or methodologies.

5.2.1.1 The Software Requirements Specification shall contain: [SWE-109]

a. System overview.

b. CSCI requirements:

(1) Functional requirements.

(2) Required states and modes.

(3) External interface requirements.

(4) Internal interface requirements.

(5) Internal data requirements.

(6) Adaptation requirements (data used to adapt a program to a given installation site or to given conditions in its operational environment).

(7) Safety requirements.

(8) Performance and timing requirements.

(9) Security and privacy requirements.

(10) Environment requirements.

(11) Computer resource requirements:

(a) Computer hardware resource requirements, including utilization requirements.

(b) Computer software requirements.

(c) Computer communications requirements.

(12) Software quality characteristics.

(13) Design and implementation constraints.

(14) Personnel-related requirements.

(15) Training-related requirements.

(16) Logistics-related requirements.

(17) Packaging requirements.

(18) Precedence and criticality of requirements.

c. Qualification provisions (e.g., demonstration, test, analysis, inspection).

d. Bidirectional requirements traceability.

e. Requirements partitioning for phased delivery.

f. Testing requirements that drive software design decisions (e.g., special system level timing requirements/checkpoint restart).

g. Supporting requirements rationale.

5.2.2 Software Data Dictionary.

5.2.2.1 The Software Data Dictionary shall include: [SWE-110]

a. Channelization data (e.g., bus mapping, vehicle wiring mapping, hardware channelization).

b. Input/Output (I/O) variables.

c. Rate group data.

d. Raw and calibrated sensor data.

e. Telemetry format/layout and data.

f. Data recorder format/layout and data.

g. Command definition (e.g., onboard, ground, test specific).

h. Effecter command information.

i. Operational limits (e.g., maximum/minimum values, launch commit criteria information).

5.2.3 Software Design Description.

The Software Design Description describes the design of a CSCI. It describes the CSCI-wide design decisions, the CSCI architectural design, and the detailed design needed to implement the software.

5.2.3.1 The Software Design Description shall include: [SWE-111]

a. CSCI-wide design decisions/trade decisions.

b. CSCI architectural design.

c. CSCI decomposition and interrelationship between components:

(1) CSCI components:

(a) Description of how the software item satisfies the software requirements, including algorithms, data structures, and functional decomposition.

(b) Software item I/O description.

(c) Static/architectural relationship of the software units.

(d) Concept of execution, including data flow, control flow, and timing.

(e) Requirements, design and code traceability.

(f) CSCI's planned utilization of computer hardware resources.

(2) Rationale for software item design decisions/trade decisions including assumptions, limitations, safety and reliability related items/concerns or constraints in design documentation.

(3) Interface design.

Note: The documentation of the architectural design of a software system identifies and describes the architectural elements of the software, the external interfaces, and the interfaces between elements. The description includes element responsibilities (constraints on inputs and guarantees on outputs), and constraints on how the elements interact (such as message and data sharing protocols). The architectural design documentation includes multiple views of the architecture and identifies and supports the evaluation of the key quality attributes of the planned software product. The key quality attributes of the software will depend on the mission in which the software is to be used and the manner in which it is to be developed and deployed. They will usually include: performance, availability, maintainability, modifiability, security, testability and usability (operability.)

5.2.4 Interface Design Description.

The Interface Design Description describes the interface characteristics of one or more systems, subsystems, Hardware Configuration Item (HWCI's), CSCI's, manual operations, or other system components. An interface design description may describe any number of interfaces.

5.2.4.1 The Interface Design Description shall include: [SWE-112]

a. Priority assigned to the interface by the interfacing entity(ies).

b. Type of interface (e.g., real-time data transfer, storage-and-retrieval of data) to be implemented.

c. Specification of individual data elements (e.g., format and data content, including bit-level descriptions of data interface) that the interfacing entity(ies) will provide, store, send, access, and receive.

d. Specification of individual data element assemblies (e.g., records, arrays, files, reports) that the interfacing entity(ies) will provide, store, send, access, and receive.

e. Specification of communication methods that the interfacing entity(ies) will use for the interface.

f. Specification of protocols the interfacing entity(ies) will use for the interface.

g. Other specifications, such as physical compatibility of the interfacing entity(ies).

h. Traceability from each interfacing entity to the system or CSCI requirements addressed by the entity's interface design, and traceability from each system or CSCI requirement to the interfacing entities that address it.

i. Interface compatibility.

j. Safety-related interface specifications and design features.

5.2.5 Software Change Request/Problem Report.

5.2.5.1 The Software Change Request/Problem Report shall contain: [SWE-113]

a. Identification of the software item.

b. Description of the problem or change to enable problem resolution or justification for and the nature of the change, including: assumptions/ constraints and change to correct software error.

c. Originator of Software Change Request/Problem Report and originator's assessment of priority/severity.

d. Description of the corrective action taken to resolve the reported problem or analysis and evaluation of the change or problem, changed software configuration item, schedules, cost, products, or test.

e. Life-cycle phase in which problem was discovered or in which change was requested.

f. Approval or disapproval of Software Change Request/Problem Report.

g. Verification of the implementation and release of modified system.

h. Date problem discovered.

i. Status of problem.

j. Identify any safety-related aspects/considerations/ impacts associated with the proposed change and/or identified problem.

k. Configuration of system and software when problem is identified (e.g., system/software configuration identifier or list of components and their versions).

l. Any workaround to the problem that can be used while a change is being developed or tested.

Note: The Software Change Request/Problem Report provides a means for identifying and recording the resolution to software anomalous behavior, process non-compliance with plans and standards, and deficiencies in life-cycle data or for identifying and recording the implementation of a change or modification in a software item.

5.2.6 Software Test Procedures.

The Software Test Procedures describe the test preparations, test cases, and test procedures to be used to perform qualification testing of a CSCI or a software system or subsystem.

5.2.6.1 The Software Test Procedures shall contain: [SWE-114]

a. Test preparations, including hardware and software.

b. Test descriptions, including:

(1) Test identifier.

(2) System or CSCI requirements addressed by the test case.

(3) Prerequisite conditions.

(4) Test input.

(5) Instructions for conducting procedure.

(6) Expected test results, including assumptions and constraints.

(7) Criteria for evaluating results.

c. Requirements traceability.

d. Identification of test configuration.

5.2.7 Software User Manual.

The Software User Manual defines user instructions for the software.

5.2.7.1 The Software User Manual shall contain: [SWE-115]

a. Software summary, including: application, inventory, environment, organization, overview of operation, contingencies, alternate states, and modes of operation, security, privacy, assistance, and problem reporting.

b. Access to the software: first-time user of the software, initiating a session, and stopping and suspending work.

c. Processing reference guide: capabilities, conventions, processing procedures, related processing, data back up, recovery from errors, malfunctions, emergencies, and messages.

d. Assumptions, limitations, and safety-related items/concerns or constraints.

e. Information that is unique or specific for each version of the software (e.g., new and modified features, new and modified interfaces).

5.2.8 Software Version Description.

The Software Version Description identifies and describes a software version consisting of one or more CSCIs (including any open source software). The description is used to release, track, and control a software version.

5.2.8.1 The Software Version Description shall identify and provide: [SWE-116]

a. Full identification of the system and software (e.g., numbers, titles, abbreviations, version

numbers, and release numbers).

b. Executable software (e.g., batch files, command files, data files, or other software needed to install the software on its target computer).

c. Software life-cycle data that defines the software product.

d. Archive and release data.

e. Instructions for building the executable software, including, for example, the instructions and data for compiling and linking and the procedures used for software recovery, software regeneration, testing, or modification.

f. Data integrity checks for the executable object code and source code.

g. Software product files (any files needed to install, build, operate, and maintain the software).

h. Open change requests and or problem reports, including any workarounds.

i. Change requests and/or problem reports implemented in the current software version since the last Software Version Description was published.

5.3 Software Reports

5.3.1 Software Metrics Report.

The Software Metrics Report provides data to the project for the assessment of software cost, technical, and schedule progress. The Software Metrics Report shall contain as a minimum the following information tracked on a CSCI basis: [SWE-117]

a. Software progress tracking measures.

b. Software functionality measures.

c. Software quality measures.

d. Software requirement volatility.

e. Software characteristics.

Note: An example set of software progress tracking measures that meet 5.3.1.a include, but are not limited to:

a. Software resources such as budget and effort (planned vs. actual).

b. Software development schedule tasks (e.g., milestones) (planned vs. actual).

c. Implementation status information (e.g., number of computer software units in design phase, coded, unit tested, and integrated into computer software configuration item versus planned).

d. Test status information (e.g., number of tests developed, executed, passed).

e. Number of replans/baselines performed.

Note: An example set of software functionality measures that meet 5.3.1.b include, but are not limited to:

a. Number of requirements included in a completed build/release (planned vs. actual).

b. Function points (planned vs. actual).

Note: An example set of software quality measures that meet 5.3.1.c include, but are not limited to:

a. Number of software Problem Reports/Change Requests (new, open, closed, severity).

b. Review of item discrepancies (open, closed, and withdrawn).

c. Number of software peer reviews/inspections (planned vs. actual).

d. Software peer review/inspection information (e.g., effort, review rate, defect data).

e. Number of software audits (planned vs. actual).

f. Software audit findings information (e.g., number and classification of findings).

g. Software risks and mitigations.

h. Number of requirements verified or status of requirements validation.

i. Results from static code analysis tools.

Note: An example set of software requirement volatility measures that meet 5.3.1.d include, but are not limited to:

a. Number of software requirements.

b. Number of software requirements changes (additions, modifications, deletions) per month.

c. Number of "to be determined" items.

Note: An example set of software characteristics that meet 5.3.1.e include, but are not limited to:

a. Project name.

b. Language.

c. Software domain (flight software, ground software, Web application).

d. Number of source lines of code by categories (e.g., new, modified, reuse) (planned vs. actual).

e. Computer resource utilization in percentage of capacity.

Note: Other information may be provided at the supplier's discretion to assist in evaluating the cost, technical, and schedule performance; e.g., innovative processes and cost reduction initiatives.

5.3.2 Software Test Report.

The Software Test Report is a record of the qualification testing performed on a CSCI, a software system or subsystem, or other software-related item.

5.3.2.1 The Software Test Report shall include: [SWE-118]

a. Overview of the test results:

(1) Overall evaluation of the software as shown by the test results.

(2) Remaining deficiencies, limitations, or constraints detected by testing (e.g., including description of the impact on software and system performance, the impact a correction would have on software and system design, and recommendations for correcting the deficiency, limitation, or constraint).

(3) Impact of test environment.

b. Detailed test results:

(1) Project-unique identifier of a test and test procedure(s).

(2) Summary of test results (e.g., including requirements verified).

(3) Problems encountered.

(4) Deviations from test cases/procedures.

c. Test log:

(1) Date(s), time(s), and location(s) of tests performed.

(2) Test environment, hardware, and software configurations used for each test.

(3) Date and time of each test-related activity, the identity of the individual(s) who performed the activity, and the identities of witnesses, as applicable.

d. Rationale for decisions.

5.3.3 Software Peer Review/Inspection Report.

5.3.3.1 The Software Peer Review/Inspection Report shall include: [SWE-119]

a. Identification information (including item being reviewed/inspected, review/inspection type (e.g., requirements inspection, code inspection, etc.) and review/inspection time and date).

b. Summary on total time expended on each software peer review/inspection (including total hour summary and time participants spent reviewing/inspecting the product individually).

c. Participant information (including total number of participants and participant's area of expertise).

d. Total number of defects found (including the total number of major defects, total number of minor defects, and the number of defects in each type such as accuracy, consistency, completeness).

e. Peer review/inspection results summary (i.e., pass, re-inspection required).

f. Listing of all review/inspection defects.

Chapter 6: Tailoring, Engineering Technical Authority, and Compliance Measurement

This NPR provides software engineering requirements to be applied throughout NASA and its contractor community. To accommodate the wide variety of software systems and subsystems, application of these requirements to specific projects will be evaluated and implemented for each project, with alternate acceptable requirements being applied where justified and approved. To effectively maintain control over the application of requirements in this NPR and to ensure proposed variants from specific requirements are appropriately mitigated, NASA has established an Engineering Technical Authority. Waivers and deviations from requirements in this NPR are governed by the following requirements as well as ones established in NPD 1000.3 and NPR 7120.5 for all of the Agency's investment areas. The Engineering Technical Authority (i.e., NASA Headquarters Chief Engineer, Center Director, or designee) for each requirement in this NPR is documented in the last column of Appendix D: Requirements Mapping Matrix. NASA Headquarters' Chief, Safety and Mission Assurance has co-approval on any waiver or deviation decided at the Headquarters level that involves safety-critical software. NASA Headquarters Chief Medical Officer has co-approval on any waiver or deviation decided at the Headquarters level that involves software with health and medical implications. Waivers and deviations decided at the Center level are to follow similar protocol when software safety critical or health and medical issues are involved.

6.1 Tailoring of Requirements

NASA Centers are expected to establish rigorous software engineering practices for software developed or acquired by projects managed at those Centers. In cases where these Center practices are formal, enforced, and meet or exceed the requirements of this NPR, the requirements allocated to "projects" in this NPR and the data items in Chapter 5 can be replaced by specific Center-level requirements approved by the designated Engineering Technical Authority.

6.1.1 For those cases in which a Center or project desires a general exclusion from requirement(s) in this NPR or desires to generically apply specific alternate requirements that do not meet or exceed the requirements of this NPR, the requester shall submit a waiver for those exclusions or alternate requirements for approval by the NASA Headquarters' Chief Engineer with appropriate justification. [SWE-120]

Note: This type of waiver (which is approved by the NASA Headquarters' Chief Engineer) is for generic/blanket relief from a requirement for a Center, Center organization, or multiple projects over an extended time. Generic/blanket waivers are not to be confused with normal waivers that address relief from a requirement on a single project or in a specific instance (which can be approved at the Center level if so specified in last column of Appendix D).

6.1.2 Where approved, the requesting Center or project shall document the approved alternate requirement in the procedure controlling the development, acquisition, and/ or deployment of the affected software. [SWE-121]

6.2 Designation of Engineering Technical Authority(s)

6.2.1 The designated Engineering Technical Authority(s) for requirements in this NPR, which can be waived at the Center level, shall be approved by the Center Director. [SWE-122]

Note: The designated Engineering Technical Authority(s) for this NPR is a recognized NASA software engineering expert. Typically, Center Directors designate an Engineering Technical Authority for software from their engineering organization for software classes A through E, from the NASA Headquarters' Office of the Chief Information Officer (CIO) for Class F, and from their Center CIO organization for classes G through H. The designation of Engineering Technical Authority(s) is documented in the Center Technical Authority Implementation Plan. Please refer to Appendix D (last column) for requirements that can be waived at the Center level.

[Requirement number SWE-123 is reserved]

6.3 Compliance

6.3.1 The designated Center Engineering Technical Authority(s) for this NPR shall comply with NASA Headquarters' Office of the Chief Engineer's direction on roles and responsibilities for Engineering Technical Authority. [SWE-124]

Note: Top-level direction on Engineering Technical Authority for all of the Agency's investment areas (including activities performed under NPR 7120.7 and NPR 7120.8) is documented in NPR 7120.5 and elaborated for software in this NPR.

6.3.2 Each project with software components shall maintain a compliance matrix against requirements in this NPR, including those delegated to other parties or accomplished by contract vehicles. [SWE-125]

Note: The compliance matrix documents the requirements that the project plans to meet and any approved waivers or deviations. The compliance matrix and compliance matrix updates should be submitted to the designated Center level Engineering Technical Authority for their records.

6.3.3 The Engineering Technical Authority(s) for this NPR shall consider the following information when assessing waivers and deviations from requirements in this NPR: [SWE-126]

a. The NASA software inventory data on the project.

b. The classification of systems and subsystems containing software, as defined in Appendix E.

c. Applicable Center-level software directives that meet the intent of this NPR.

d. Applicable contractor and subcontractor software policies and procedures that meet the intent of this NPR.

e. Potential impacts to NASA missions.

f. Potential impacts to health, medical concerns, or safety.

6.3.4 Centers and projects shall fully comply with the "shall" statements in this NPR that are marked with an "X" in Appendix D consistent with their software classification. [SWE-139]

Note: If the requirement is marked with a "SO" in appendix D, the project is required to meet the requirement to the extent necessary to satisfy the safety critical aspects of the software.

6.3.5 When the requirement and software class are marked with a "P (Center)," Centers and projects shall meet the requirement with an approved non-null subset of the "shall" statement (or approved alternate) for that specific requirement. [SWE 140]

Note: The use of partial Center (i.e., "P (Center)") requirements allows for local adaptations to suit

Center and application unique needs. Although the NASA Headquarters' Office of the Chief Engineer is the review and concurrence authority for the Center defined "P (Center)" requirements, this approval does not constitute a deviation nor a waiver. The project or Center is responsible for flowing down all applicable NPR 7150.2 requirements, including "P (Center)" requirements, to contracted software development activities. "P (Center)" requirements are typically documented in Center-level directives.

6.3.6 The NASA Headquarters' Office of the Chief Engineer shall review and have concurrence approval when a Center defines subsets of requirements denoted by "P (Center)" in the Requirements Mapping Matrix in Appendix D for the indicated classes of software. [SWE-127]

6.3.7 The Center-level Engineering Technical Authority shall keep records of projects' compliance matrices, waivers, and deviations against this NPR. [SWE-128]

6.3.8 The NASA Headquarters' Office of the Chief Engineer shall authorize appraisals against selected requirements in this NPR (including NASA Headquarters' Office of the Chief Engineer approved subsets and alternative sets of requirements) to check compliance. [SWE-129]

Appendix A. Definitions

A.1 **Accuracy**. The difference between a parameter or variable (or a set of parameters or variables) within a model, simulation, or experiment and the true value or the assumed true value (Definition from source document: NASA-STD-7009, Standard for Models and Simulations.).

A.2 **Accredit**. The official acceptance of a software development tool, model, or simulation, (including associated data) to use for a specific purpose.

A.3 **Analysis**. The post-processing or interpretation of the individual values, arrays, files of data, or execution information.

A.4 **Contracted Software**. Software created for a project by a contractor or subcontractor.

A.5 **Data**. Information for computer processing (e.g., numbers, text, images, and sounds in a form that is suitable for storage in or processing by a computer).

A.6 **Deviation**. A documented authorization releasing a program or project from meeting a requirement before the requirement is put under configuration control at the level the requirement will be implemented.

A.7 **Embedded Software**. Software that is part of a larger system and performs some of the requirements of that system (Definition from source document: ISO/IEC 24765:2009 Systems and Software Engineering Vocabulary.).

A.8 **Establish and Maintain**. The responsible project, organization, or individual must formulate, document, use/deploy, and keep current the object (usually a document, requirement, process, or policy).

A.9 **Glueware**. Software created to connect the off-the-shelf software/reused software with the rest of the system. It may take the form of "adapters" that modify interfaces or add missing functionality, "firewalls" that isolate the off-the-shelf software, or "wrappers" that check inputs and outputs to the off-the-shelf software and may modify to prevent failures.

A.10 **Government Off-The-Shelf (GOTS) Software**. This refers to Government-created software, usually from another project. The software was not created by the current developers (see software reuse). Usually, source code is included and documentation, including test and analysis results, is available. That is, the government is responsible for the GOTS software to be incorporated into another system. (Definition from source document: NASA-GB-8719.13, NASA Software Safety Guidebook.)

A.11 **Insight**. Surveillance mode requiring the monitoring of customer-identified metrics and contracted milestones. Insight is a continuum that can range from low intensity, such as reviewing quarterly reports, to high intensity, such as performing surveys and reviews. (Definitions from source document: NPR 8735.2, Management of Government Safety and Mission Quality Assurance Surveillance Functions for NASA Contracts.)

A.12 **Independent Verification and Validation (IV&V)**. Verification and validation performed by an organization that is technically, managerially, and financially independent of the development organization (ISO/IEC 24765:2008 systems and software engineering vocabulary).

A.13 **Legacy/Heritage**. Software products (architecture, code, requirements) written specifically for one project and then, without prior planning during its initial development, found to be useful on

other projects. See software reuse.

A.14 **Major Engineering/Research Facility**. Used in this document to show research, development, test, or simulation facilities representing a significant NASA investment which contains software that supports programs and projects managed under NPR 7120.5, NPR 7120.7, or NPR 7120.8. Examples include: high-fidelity, motion-base flight simulator facilities (e.g., Vertical Motion Simulator at Ames), wind tunnels (e.g., National Transonic Facility at LaRC), vacuum chambers (e.g., Space Power Facility at GRC/Plum Brook), air traffic control facilities (e.g., North Texas Research Station), and engine test stands (e.g., J-2X test stand at Stennis Space Center). Related major facility designations are contained in NPR 8800.15, Section 3.8.1, Designation of Major Facilities.

A.15 **Mathematical Model**. The mathematical equations, boundary values, initial conditions, and modeling data needed to describe the conceptual model (ASME V&V 10). (Definition from source document: NASA-STD-7009, Standard for Models and Simulations.)

A.16 **Mission Critical**. Item or function that must retain its operational capability to assure no mission failure (i.e., for mission success). (Definition from source document: NPR 8715.3, NASA General Safety Manual Program Requirements.)

A.17 **Model**. A description or representation of a system, entity, phenomena, or process. (Definition from source document: NASA-STD-7009, Standard for Models and Simulations.) Only for the purpose of this document, the term "model" refers to only those models which are implemented in software.

A.18 **Modified Off-The-Shelf (MOTS) Software**. When COTS, legacy/heritage software is reused, or heritage software is changed, the product is considered "modified." The changes can include all or part of the software products and may involve additions, deletions, and specific alterations. An argument can be made that any alterations to the code and/or design of an off-the-shelf software component constitutes "modification," but the common usage allows for some percentage of change before the off-the-shelf software is declared to be MOTS software. This may include the changes to the application shell and/or glueware to add or protect against certain features and not to the off-the-shelf software system code directly. See off-the-shelf.

A.19 **Off-The-Shelf Software**. Software not developed in-house or by a contractor for the specific project now underway. The software is generally developed for a different purpose than the current project.

A.20 **Operational Software**. Software that has been accepted and deployed, delivered to its customer, or is deployed in its intended environment.

A.21 **Primary Mission Objectives**. Outcomes expected to be accomplished which are closely associated with the reason the mission was proposed, funded, developed, and operated (e.g., objectives related to top-level requirements or their flow down).

A.22 **Process Asset Library**. A collection of process asset holdings that can be used by an organization or project. (Definition from source document: CMMI® for- Systems Engineering/Software Engineering/Integrated Product and Process Development Supplier Sourcing.)

A.23 **Program**. A strategic investment by a Mission Directorate or Mission Support Office that has a defined architecture and/or technical approach, requirements, funding level, and a management structure that initiates and directs one or more projects. A program defines a strategic direction that the Agency has identified as critical.

A.24 **Project**. A specific investment having defined goals, objectives, requirements, life-cycle cost,

a beginning, and an end. A project yields new or revised products or services that directly address NASA's strategic needs. They may be performed wholly in-house; by Government, industry, academia partnerships, or through contracts with private industry.

A.25 **Reuse**. See software reuse.

A.26 **Risk Management**. An organized, systematic decision-making process that efficiently identifies, analyzes, plans, tracks, controls, communicates, and documents risk to increase the likelihood of achieving program/project goals. (Definition from source document: NPR 8735.2, Management of Government Quality Safety and Mission Assurance Surveillance Functions for NASA Contracts.) Risk management includes Risk-Informed Decision Making and Continuous Risk Management in an integrated framework. This is done to foster proactive risk management, to better inform decision making through better use of risk information, and then to more effectively manage implementation risks by focusing the Continuous Risk Management process on the baseline performance requirements emerging from the Risk-Informed Decision-Making process. (Definition from source document: NPR 8000.4, Agency Risk Management Procedural Requirements.)

A.27 **Scripts**. A sequence of automated computer commands embedded in a program that tells the program to execute a specific procedure (e.g., files with monitoring, logic, or commands used by software to automate a process or procedure).

A.28 **Sensitivity Analysis**. The study of how the variation in the output of a model can be apportioned to different sources of variation in the model input and parameters. (Definition from source document: NASA-STD-7009, Standard for Models and Simulations.)

A.29 **Simulation**. The imitation of the characteristics of a system, entity, phenomena, or process using a computational model. (Definition from source document: NASA-STD-7009, Standard for Models and Simulations.) Only for the purpose of this document, the term "simulation" refers to only those simulations which are implemented in software.

A.30 **Software**. Computer programs, procedures, scripts, rules, and associated documentation and data pertaining to the development and operation of a computer system. Software includes programs and data. This also includes COTS, GOTS, MOTS, reused software, auto generated code, embedded software, firmware, and open source software components.

A.31 **Software Architecture**. The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the properties of those components, and the relationships between them. The term also refers to documentation of a system's software architecture. Documenting software architecture facilitates communication between stakeholders, documents early decisions about high-level design, and allows reuse of design components and patterns between projects.

A.32 **Safety-Critical Software**. See definition in NASA-STD-8719.13, Software Safety Standard.

A.33 **Software Engineering**. The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software: that is, the application of engineering to software. (Definition from source document: IEEE 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology.)

A.34 **Software Peer Review/Inspection**. A visual examination of a software product to detect and identify software anomalies, including errors and deviations from standards and specifications (IEEE 1028-2008 IEEE Standard for Software Reviews and Audits). Guidelines for software peer reviews/inspections are contained in NASA-STD-2202-93, NASA Software Formal Inspection Standard.

A.35 **Software Reuse**. A software product developed for one use but having other uses or one developed specifically to be usable on multiple projects or in multiple roles on one project. Examples include, but are not limited to, COTS products, acquirer-furnished software products, software products in reuse libraries, and pre-existing developer software products. Each use may include all or part of the software product and may involve its modification. This term can be applied to any software product (such as, requirements and architectures), not just to software code itself. Often this is software previously written by an in-house development team and used on a different project. GOTS software would come under this category if the product is supplied from one Government project to another Government project. (Definition from source document: NASA-GB-8719.13, NASA Software Safety Guidebook.)

A.36 **Software Validation**. Confirmation that the product, as provided (or as it will be provided), fulfills its intended use. In other words, validation ensures that "you built the right thing."

A.37 **Software Verification**. Confirmation that work products properly reflect the requirements specified for them. In other words, verification ensures that "you built it right."

A.38 **Static Analysis**. The process of evaluating a system or component based on its form, structure, content, or documentation. (Definition from source document: ISO/IEC 24765:2008 systems and software engineering vocabulary.)

A.39 **Subsystem**. A secondary or subordinate system within a larger system. (Definition from source document: ISO/IEC 24765:2008 systems and software engineering vocabulary.)

A.40 **System**. The combination of elements that function together to produce the capability required to meet a need. The elements include hardware, software, equipment, facilities, personnel, processes, and procedures needed for this purpose. (Definition from source document: NPR 7123.1A NASA Systems Engineering Processes and Requirements.)

A.41 **Uncertainty**. (1) The estimated amount or percentage by which an observed or calculated value may differ from the true value. (2) A broad and general term used to describe an imperfect state of knowledge or a variability resulting from a variety of factors including, but not limited to, lack of knowledge, applicability of information, physical variation, randomness or stochastic behavior, indeterminacy, judgment, and approximation (adapted from NPR 8715.3B, NASA General Safety Program Requirements).

A.42 **Waiver**. A documented authorization intentionally releasing a program or project from meeting a requirement after the requirement is put under configuration control at the level the requirement will be implemented.

A.43 **Wrapper**. See glueware definition.

Appendix B. Acronyms

CAM/CAD	Computer-Aided Design and Computer-Aided Manufacturing
CDR	Critical Design Review
CMMI®	Capability Maturity Model Integration
CMMI-DEV	Capability Maturity Model® Integration® (CMMI®) for Development
CMU	Carnegie Mellon University
COTS	Commercial-Off-The-Shelf
CSCI	Computer Software Configuration Item
EDL	Entry, Descent, and Landing
EVA	Extra Vehicular Activity
FAR	Federal Acquisition Regulation
GOTS	Government-Off-The-Shelf
HQ CE	NASA Headquarters' Chief Engineer
HQ CSMA	NASA Headquarters' Chief Safety and Mission Assurance
HWCI	Hardware Configuration Item
I/O	Input/Output
IEEE	Institute of Electrical and Electronics Engineers
IPEP	IV&V Project Execution Plan
ITMRA	Information Technology Management Reform Act
IV&V	Independent Verification and Validation
MOTS	Modified-Off-The-Shelf
NPD	NASA Policy Directive
NPR	NASA Procedural Requirements
PDR	Preliminary Design Review

R&T	Research and Technology
SCAMPI	Standard CMMI® Appraisal Method for Process Improvement
SEI	Software Engineering Institute
SMA	Safety and Mission Assurance
SRR	Software Requirements Review
SWE	Software Engineering
TA	Technical Authority

Appendix C. References

a. NASA-STD-2202-93, NASA Software Formal Inspection Standard.

b. NASA-GB-8719.13, NASA Software Safety Guidebook.

c. NASA-STD-3000, Volumes I-II, Man-Systems Integration Standards.

d. NASA-STD-7009, Standard for Models and Simulations.

e. CMU/SEI-2002-TR-012, Capability Maturity Model Integration (CMMI) for Systems Engineering/Software Engineering/Integrated Product and Process Development/Supplier Sourcing.

f. IEEE 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology.

g. IEEE 1028-2008 IEEE Standard for Software Reviews and Audits.

h. IEEE 1012-2004 IEEE Standard for Software Verification and Validation.

i. ISO/IEC 24765:2008 Systems and Software Engineering Vocabulary.

j. NASA Software Engineering Web site, http://software.nasa.gov/

k. NASA Engineering Network Web site, http://nen.nasa.gov/portal/site/llis/OCE/

l. NASA Software Process Asset Library (PAL) Web site, http://swpal.nasa.gov/

m. NASA IV&V Management System, http://www.nasa.gov/centers/ivv/ims/home/index.html

n. 36 CFR Part 1194 Electronic and Information Technology Accessibility Standards.

Appendix D. Requirements Mapping Matrix

Section of NPR	Requirement Descriptor**	SWE #	Responsibility	Class A OR Class A and Safety Critical Note 2	Class B OR Class B and Safety Critical Note 2	Classes C thru E and Safety Critical Note 2	Class C and NOT Safety Critical	Class D and NOT Safety Critical	Class E and NOT Safety Critical	Class F	Class G	Class H	Technical Authority for NPR 7150.2 by Requirement (Note 3)
Preface	Effective date	1	General	X	X	X	X	X	X	X	X	X	HQ CE
Organizational Capability	Agency software initiative	2	NASA Chief Engineer	X	X	X	X	X	X	X	X	X	HQ CE
	Center plan	3	Each Center	X	X	X	X	X	P (Center)	X	X	P (Center)	HQ CE
	Benchmark	4	NASA Chief Engineer	X	X	X	X	X	X	X	X	X	HQ CE
	Software processes	5	Each Center	X	X	X	X	X	P (Center)	X	X	P (Center)	HQ CE
	List of agency's programs & projects containing software	6	NASA Chief Engineer & Chief, Safety Mission Assurance	X	X	X	X	X		X	X		HQ CE/ CSMA
SW Life-Cycle Planning	Software plans	13	Project	X	X	X	X	P (Center)	P (Center)	X	P (Center)		HQ CE
	Execute planning	14	Project	X	X	X	X	X	X	X	P (Center)		Center Director*
	Cost estimation	15	Project	X	X	X	X	P (Center)	P (Center)	X	P (Center)		Center Director*
	Schedule	16	Project	X	X	X	X	P (Center)		X	P (Center)		Center Director*
	Training	17	Project	X	X	X	X			X	P (Center)		Center Director*
	Reviews	18	Project	X	X	X	X	X		X	P (Center)		Center Director*
	model	19	Project	X	X	X	X	(Center)		OTS)	(Center)		Director*

Section of NPR	Requirement Descriptor**	SWE #	Responsibility	Class A OR Class A and Safety Critical Note 2	Class B OR Class B and Safety Critical Note 2	Classes C thru E and Safety Critical Note 2	Class C and NOT Safety Critical	Class D and NOT Safety Critical	Class E and NOT Safety Critical	Class F	Class G	Class H	Technical Authority for NPR 7150.2 by Requirement (Note 3)
	Software classification	20	Project	X	X	X	X	X	X	X	X	X	HQ CE
SW Life-Cycle Planning	Software classification changes	21	Project	X	X	X	X	X	X	X	X	X	Center Director*
	Software assurance	22	Project	X	X	X	X	X		X	X		HQ CE/ CSMA
	Software safety	23	Project	X	X	X	X	X	X	X	X	X	HQ CE/ CSMA
	Plan tracking	24	Project	X	X	X	X	P (Center)		X	P (Center)		Center Director*
	Corrective action	25	Project	X	X	X (SO if D-E)	X			X	P (Center)		Center Director*
	Changes	26	Project	X	X	X	X			X	P (Center)		Center Director*
Off The Shelf (OTS) SW	COTS, GOTS, MOTS, etc.	27	Project	X	X	X	X			X	P (Center)		Center Director*
Verification & Validation	Verification planning	28	Project	X	X	X	X	P (Center)		X	P (Center)		Center Director*
	Validation planning	29	Project	X	X	X	X	P (Center)		X	P (Center)		Center Director*
	Verification results	30	Project	X	X	X	X	X		X	P (Center)		Center Director*
	Validation results	31	Project	X	X	X	X	X		X	P (Center)		Center Director*
Project Formulation	CMMI levels for class A, B, and C software	32	Project	X	X (Note 1)	P (Center) for Class C only	P (Center)						HQ CE
	Acquisition Assessment	33	Project	X	X	X	X	X		X	X		Center Director*
	criteria	34	Project	X	X	X	X	X		X	X		Director*

Section of NPR	Requirement Descriptor**	SWE #	Responsibility	Class A OR Class A and Safety Critical Note 2	Class B OR Class B and Safety Critical Note 2	Classes C thru E and Safety Critical Note 2	Class C and NOT Safety Critical	Class D and NOT Safety Critical	Class E and NOT Safety Critical	Class F	Class G	Class H	Technical Authority for NPR 7150.2 by Requirement (Note 3)
	Supplier selection	35	Project	X	X	X (SO if D-E)	X			X	X	P (Center)	Center Director*
	Software processes	36	Project	X	X	X	X	P (Center)		X (not OTS)	P (Center)		Center Director*
Project Formulation	Software Milestones	37	Project	X	X	X	X	P (Center)		X (not OTS)	P (Center)		Center Director*
	Acquisition planning	38	Project	X	X	X	X	X		X (not OTS)	P (Center)		Center Director*
Government Insight	Insight into software activities	39	Project	X	X	P (Center) + SO	P (Center)	P (Center)		X	P (Center)		Center Director*
	Access to software products	40	Project	X	X	P (Center) + SO	P (Center)			X	P (Center)		Center Director*
	Open source notification	41	Project	X	X	X (SO if D-E)	X			X	P (Center)		Center Director*
	Electronic access to Source code	42	Project	X	X	X (SO if D-E)	X			X	P (Center)		Center Director*
Supplier Monitoring	Track change request	43	Project	X	X	P (Center) + SO	P (Center)			X	P (Center)		Center Director*
	Software measurement data	44	Project	X	X	X	X	P (Center)		X	P (Center)		Center Director*
	Joint audits	45	Project	X	X	X (SO if D-E)	X			X	P (Center)		Center Director*
	Software schedule	46	Project	X	X	X	X	X		X	P (Center)		Center Director*
	Traceability data	47	Project	X	X	X (SO if D-E)	X			X	P (Center)		Center Director*
	Solicitation	48	Project	X	X	X	X	P (Center)		X	P (Center)		Center Director*
Software	Document	49	Project	X	X	X	X	X	(Center)	X	X		Director*

Section of NPR	Requirement Descriptor**	SWE #	Responsibility	Class A OR Class A and Safety Critical Note 2	Class B OR Class B and Safety Critical Note 2	Classes C thru E and Safety Critical Note 2	Class C and NOT Safety Critical	Class D and NOT Safety Critical	Class E and NOT Safety Critical	Class F	Class G	Class H	Technical Authority for NPR 7150.2 by Requirement (Note 3)
Requirements Development	Software requirements	50	Project	X	X	X	X	X		X	X		Center Director*
	Flow-down & derived requirements	51	Project	X	X	X	X			X (not OTS)	P (Center)		Center Director*
Software Requirements Development	Bidirectional traceability	52	Project	X	X	P (Center) + SO	P (Center)			X (not OTS)	P (Center)		Center Director*
Software Requirements Management	Manage requirements change	53	Project	X	X	X	X	X	P (Center)	X	X		Center Director*
	Corrective action	54	Project	X	X	X				X	X		Center Director*
	Requirements validation	55	Project	X	X	X	X			X	X		Center Director*
Software Design	Document design	56	Project	X	X	X	X			X (not OTS)	X (not OTS)		Center Director*
	Software architecture	57	Project	X	X	P (Center) + SO	P (Center)	P (Center)		X (not OTS)	X (not OTS)		Center Director*
	Detailed design	58	Project	X	X	SO				X (not OTS)	X (not OTS)		Center Director*
	Bidirectional traceability	59	Project	X	X	SO				X (not OTS)	X (not OTS)		Center Director*
Software Implementation	Design into code	60	Project	X	X	X	X	X		X (not OTS)	X (not OTS)		Center Director*
	Coding standards	61	Project	X	X	X				X (not OTS)	X (not OTS)		Center Director*
	Unit test	62	Project	X	X	X	X	P (Center)		X (not OTS)	X (not OTS)		Center Director*
	Version description	63	Project	X	X	X	X	P (Center)		X (not OTS)	X (not OTS)		Center Director*
	Bidirectional traceability	64	Project	X	X	SO				X (not OTS)	X (not OTS)		Center Director*
Software Testing		65	Project	X	X	X	X	(Center)		X	(Center)		Director*

Section of NPR	Requirement Descriptor**	SWE #	Responsibility	Class A OR Class A and Safety Critical Note 2	Class B OR Class B and Safety Critical Note 2	Classes C thru E and Safety Critical Note 2	Class C and NOT Safety Critical	Class D and NOT Safety Critical	Class E and NOT Safety Critical	Class F	Class G	Class H	Technical Authority for NPR 7150.2 by Requirement (Note 3)
	reports												
	Perform testing	66	Project	X	X	X	X	X	P (Center) (Note 6)	X	P (Center)		Center Director*
	Verify implementation	67	Project	X	X	X	X			X	P (Center)		Center Director*
Software Testing	Evaluate test results	68	Project	X	X	X	X	X		X	P (Center)		Center Director*
	Document defects & track	69	Project	X	X	X	X	P (Center)		X	P (Center)		Center Director*
	Models, simulations, tools	70	Project	X	X	X (if Note 5 is true)	X (if Note 5 is true)	X (if Note 5 is true)		X	P (Center)		Center Director*
	Update plans & procedures	71	Project	X	X	X	X			X	P (Center)		Center Director*
	Bidirectional traceability	72	Project	X	X	X	X	X		X	P (Center)		Center Director*
	Platform or hi-fidelity simulations	73	Project	X	X	X	X			X	P (Center)		Center Director*
Software Operations, Maintenance, and Retirement	Document maintenance plan	74	Project	X	X	X	X			X	P (Center)		Center Director*
	Plan operations, maintenance & retirement	75	Project	X	X	X	X			X	P (Center)		Center Director*
	Implement plans	76	Project	X	X	X	X			X	P (Center)		Center Director*
	Deliver software products	77	Project	X	X	X	X	X	P (Center)	X	X		Center Director*
	documentation	78	Project	X	X	X	X			OTS)	OTS)		Director*

Section of NPR	Requirement Descriptor**	SWE #	Responsibility	Class A OR Class A and Safety Critical Note 2	Class B OR Class B and Safety Critical Note 2	Classes C thru E and Safety Critical Note 2	Class C and NOT Safety Critical	Class D and NOT Safety Critical	Class E and NOT Safety Critical	Class F	Class G	Class H	Technical Authority for NPR 7150.2 by Requirement (Note 3)
Software Configuration Management	Develop configuration management plan	79	Project	X	X	X	X	X		X	P (Center)		Center Director*
	Track & evaluate changes	80	Project	X	X	X	X	P (Center)		X	P (Center)		Center Director*
Software Configuration Management	Identify software configuration items	81	Project	X	X	X	X	X		X	P (Center)	P (Center)	Center Director*
	Authorizing changes	82	Project	X	X	X				X	P (Center)	P (Center)	Center Director*
	Maintain records	83	Project	X	X	X	X			X	P (Center)	P (Center)	Center Director*
	Configuration audits	84	Project	X	X	SO				X	P (Center)		Center Director*
	Implement procedures	85	Project	X	X	X	X	P (Center)	P (Center)	X	P (Center)	P (Center)	Center Director*
Risk Management	Continuous risk management	86	Project	X	X	SO				X	P (Center)		HQ CE/ CSMA
Software Peer Reviews/ Inspections	Requirements, test plans, design & code	87	Project	X	X	X (SO if D-E)	X			X (not OTS)	P (Center)		Center Director*
	Checklist, criteria & tracking	88	Project	X	X	P (Center) + SO	P (Center)			X (not OTS)	P (Center)		Center Director*
	Basic measurements	89	Project	X	X					X (not OTS)	P (Center)		Center Director*
Software Measurement	Objectives	90	Project	X	X	(SO if D-E)	X			OTS)	(Center)		Director*

Section of NPR	Requirement Descriptor**	SWE #	Responsibility	Class A OR Class A and Safety Critical Note 2	Class B OR Class B and Safety Critical Note 2	Classes C thru E and Safety Critical Note 2	Class C and NOT Safety Critical	Class D and NOT Safety Critical	Class E and NOT Safety Critical	Class F	Class G	Class H	Technical Authority for NPR 7150.2 by Requirement (Note 3)
	Software measurement areas	91	Project	X	X	P (Center) + SO	P (Center)			X (not OTS)	P (Center)		Center Director*
	Collection & storage	92	Project	X	X	X (SO if D-E)	X			X (not OTS)	P (Center)		Center Director*
	Analyze data	93	Project	X	X	P (Center) + SO	P (Center)			X (not OTS)	P (Center)		Center Director*
	Report analysis	94	Project	X	X	P (Center) + SO	P (Center)			X (not OTS)	P (Center)		Center Director*
Software Measurement	Software measurement system	95	Each Mission Directorate	X	X	X	X			X	X		HQ CE
	Objectives & procedures	96	Each Mission Directorate	X	X	X	X			X	X		HQ CE
Best Practices	Agency process asset library	98	NASA Chief Engineer	X	X	X	X	X	X	X	X	X	HQ CE
	Identify applicable practices	99	Each Center	X	X	X	X	X	X	X	X	X	HQ CE
Training	Software engineering training	100	NASA Chief Engineer & Center Training Org.	X	X	X	X	X	X	X	X	X	HQ CE
	Software training plan	101	Each Center	X	X	X	X	X	X	X	X	X	HQ CE
Software Documentation Requirements	plan	102	Project	X	X	+ SO	(Center)	(Center)		OTS)	(Center)		Director*

Section of NPR	Requirement Descriptor**	SWE #	Responsibility	Class A OR Class A and Safety Critical Note 2	Class B OR Class B and Safety Critical Note 2	Classes C thru E and Safety Critical Note 2	Class C and NOT Safety Critical	Class D and NOT Safety Critical	Class E and NOT Safety Critical	Class F	Class G	Class H	Technical Authority for NPR 7150.2 by Requirement (Note 3)
	Software configuration management plan	103	Project	X	P (Center) + SO	P (Center) + SO	P (Center)	P (Center)		X	P (Center)		Center Director*
	Software test plan	104	Project	X	X	P (Center) + SO	P (Center)	P (Center)		X	P (Center)		Center Director*
	Software maintenance plan	105	Project	X	P (Center) + SO	SO				X	P (Center)		Center Director*
	Software assurance plan	106	Project	X	X	X	X			X	X		HQ CE/ CSMA
Software Documentation Requirements	Center software training plan	107	Center	X (1 plan per center)	X (1 plan per center)	X (1 plan per center)	X (1 plan per center)	X (1 plan per center)	P (Center)	X (1 plan per center)	X (1 plan per center)	P (Center)	HQ CE
	Center software engineering improvement plan	108	Center	X (1 plan per center)	X (1 plan per center)	X (1 plan per center)	X (1 plan per center)	X (1 plan per center)	P (Center)	X (1 plan per center)	X (1 plan per center)	P (Center)	HQ CE
	Software requirements specification	109	Project	X	X	P (Center) + SO	P (Center)	P (Center)		X	X		Center Director*
	Software data dictionary	110	Project	X	P (Center) + SO	P (Center) + SO	P (Center)			X (not OTS)	X (not OTS)		Center Director*
	Software design description	111	Project	X	X	P (Center) + SO	P (Center)	P (Center)		X (not OTS)	X (not OTS)		Center Director*
	Interface design description	112	Project	X	X	P (Center) + SO	P (Center)			X (not OTS)	X (not OTS)		Center Director*
	problem report	113	Project	X	X	+ SO	(Center)			OTS)	OTS)		Director*

Section of NPR	Requirement Descriptor**	SWE #	Responsibility	Class A OR Class A and Safety Critical Note 2	Class B OR Class B and Safety Critical Note 2	Classes C thru E and Safety Critical Note 2	Class C and NOT Safety Critical	Class D and NOT Safety Critical	Class E and NOT Safety Critical	Class F	Class G	Class H	Technical Authority for NPR 7150.2 by Requirement (Note 3)
	Software test procedures	114	Project	X	X	X	X			X	X		Center Director*
	Software users manual	115	Project	X	X	SO				X (not OTS)	X (not OTS)		Center Director*
	Software version description	116	Project	X	X	P (Center) + SO	P (Center)	P (Center)		X (not OTS)	X (not OTS)		Center Director*
	Software metrics report	117	Project	X	X	P (Center) + SO	P (Center)			X (not OTS)	X (not OTS)		Center Director*
Software Documentation Requirements	Software test report	118	Project	X	X	P (Center) + SO	P (Center)			X	P (Center)		Center Director*
	Software inspection/ peer review/ inspections	119	Project	X	X	P (Center) + SO	P (Center)			X (not OTS)	P (Center)		Center Director*
Tailoring of Requirements	Submit generic waiver request	120	Center or Project	X	X	X	X	X	X	X	X	X	HQ CE
	Document approved alternate requirements	121	Center or Project	X	X	X	X	X	X	X	X	X	HQ CE
Authority	approval	122	Center Director	X	X	X	X	X	X				HQ CE

Section of NPR	Requirement Descriptor**	SWE #	Responsibility	Class A OR Class A and Safety Critical Note 2	Class B OR Class B and Safety Critical Note 2	Classes C thru E and Safety Critical Note 2	Class C and NOT Safety Critical	Class D and NOT Safety Critical	Class E and NOT Safety Critical	Class F	Class G	Class H	Technical Authority for NPR 7150.2 by Requirement (Note 3)
Compliance	Direction for Technical Authority	124	Center level Engineering Technical Authority	X	X	X	X	X	X	X	X	X	HQ CE
	Compliance matrix	125	Project	X	X	X	X	X	X	X	X	X	HQ CE
	Considerations for waivers	126	Engineering Technical Authority	X	X	X	X	X	X	X	X	X	HQ CE
	Review of "P(Center)"	127	NASA HQ CE	X	X	X	X	X	X	X	X	X	HQ CE
Compliance	Compliance records	128	Center level Engineering Technical Authority	X	X	X	X	X	X	X	X	X	HQ CE
	Check compliance	129	NASA HQ OCE	X	X	X	X	X	X	X	X	X	HQ CE
Software Lifecycle Planning	Software safety plan	130	Project	X	X	X				X (if Safety Critical)	X (if Safety Critical)	X (if Safety Critical)	HQ CE/ CSMA
	IV&V Plan	131	IV&V Program	X (if selected for IV&V)	X (if selected for IV&V)	X (if selected for IV&V)	X (if selected for IV&V)						HQ CE/ CSMA
	Assessment	132	organization	X	X	X	X	X	X	X	X	X	CSMA

Section of NPR	Requirement Descriptor**	SWE #	Responsibility	Class A OR Class A and Safety Critical Note 2	Class B OR Class B and Safety Critical Note 2	Classes C thru E and Safety Critical Note 2	Class C and NOT Safety Critical	Class D and NOT Safety Critical	Class E and NOT Safety Critical	Class F	Class G	Class H	Technical Authority for NPR 7150.2 by Requirement (Note 3)
	Software safety determination	133	Project & SMA organization	X	X	X	X	X	X	X	X	X	HQ CE/ CSMA
	Safety-critical software requirements	134	Project	X	P (Center)	(see Note 7)							Center Director* (joint Engineering TA & SMA TA if delegated)
Software Implementation	Static analysis	135	Project	X	X	X (SO if D-E)	X						Center Director*
	Validation of software development tools	136	Project	X	X	X (if Note 4 is true)	X (if Note 4 is true)	X (if Note 4 is true)					Center Director*
Software Peer Reviews/ Inspections	Peer Review/ inspections of Software plans	137	Project	X	X	P (Center) + SO	P (Center)			X (not OTS)	P (Center)		Center Director*
Software Documentation Requirements	Software safety plan contents	138	Project	X	X	X				X (if Safety Critical)	X (if Safety Critical)	X (if Safety Critical)	HQ CE/ CSMA
Compliance	"Shall" statements in this NPR	139	Project, Center	X	X	X	X	X	X	X	X	X	HQ CE
	"P (Center)"	140	Project, Center	X	X	X	X	X	X	X	X	X	HQ CE

* Center Director or the Center Director's designed Engineering Technical Authority

** See Requirement in NPR for full description

Grey shaded box indicates that the responsibility for that requirement is not at the project level

X - "Required" - Responsible party is required to meet the requirement as written.

X (not OTS) - Project is required to meet except for off-the-shelf software.

P (Center) - Per approved Center defined process which meets a non-empty subset of the full requirement.

SO - "Safety Only" - Project is required to meet this requirement to the extent necessary to satisfy safety critical aspects of the software.

P (Center) + SO - Responsible party is required to meet this requirement to the extent necessary to satisfy safety critical aspects of the software and follow Center defined P (Center) processes for other aspects of the software.

Blank Space - Project is not required to meet the requirement.

Note 1 - For Class B software, in lieu of a CMMI rating by a development organization, the project will conduct an evaluation, performed by a qualified evaluator selected by the Center Engineering Technical Authority, of the seven process areas listed in SWE-032 and mitigate any risk, if deficient. This exception is intended to be used in those cases in which NASA wishes to purchase a product from the "best of class provider," but the best of class provider does not have the required CMMI rating. When this exception is exercised, the Center Engineering Technical Authority should be notified.

Note 2 - Note that there are additional safety requirements in NASA-STD-8719.13, NASA Software Safety Standard.

Note 3 - NASA Headquarters' Chief, Safety and Mission Assurance has co-approval on any waiver or deviation decided at the HQ level that involves safety-critical software. NASA Headquarters' Chief Medical Officer has co-approval on any waiver or deviation decided at the HQ level that involves software with health and medical implications. Waivers and deviations decided at the Center level are to follow similar protocol when software safety critical or health and medical issues are involved.

Note 4 - When software development tools are used to develop or maintain Class A, B, C, or safety-critical software.

Note 5 - When software models, simulations, and analysis tools are used to perform qualification of flight software or flight equipment.

Note 6 - No test plans are required but the project shall perform software testing.

Note 7 - This NPR does not require SWE-134 for Classes C thru E and Safety Critical; however it is highly recommended that software within this category use SWE-134 as a checklist to support the identification of safety related risks and their mitigations.

HQ CE - NASA Headquarters' Chief Engineer

HQ CSMA - NASA Headquarters' Chief, Safety and Mission Assurance

Appendix E. Software Classifications

The applicability of requirements in this NPR to specific systems and subsystems containing software is determined through the use of the NASA-wide definitions for software classes defined below and the designation of the software as safety-critical or non-safety-critical in conjunction with the Requirements Mapping Matrix in Appendix D. These definitions are based on 1) usage of the software with or within a NASA system, 2) criticality of the system to NASA's major programs and projects, 3) extent to which humans depend upon the system, 4) developmental and operational complexity, and 5) extent of the Agency's investment. Classes A – E cover engineering related software in decreasing order of applicable NPR 7150.2 requirements. Classes F through H cover business and IT software in decreasing order of applicable NPR 7120.2 requirements. Using the Requirements Mapping Matrix, the number of applicable requirements and their associated rigor are scaled back for lower software classes and software designated as non-safety-critical. Situations where a project contains separate systems and subsystems having different software classes are not uncommon.

For a given system or subsystem, software is expected to be uniquely defined within a single class. If more than one software class seems to apply, then assign the higher of the classes to the system/subsystem. Any potential discrepancies in classifying software within Classes A - E are to be resolved using the definitions and the five underlying factors listed in the previous paragraph. Engineering and Safety and Mission Assurance provide dual Technical Authority chains for resolving classification issues. The NASA Headquarters' Chief Engineer is the ultimate Technical Authority for software classification disputes concerning definitions in this NPR.

Class A: Human Rated Space Software Systems

Definition:

Human Space Flight Software Systems*: (ground and flight) developed and/or operated by or for NASA that are needed to perform a primary mission objective of human space flight and directly interacts with human space flight systems. Limited to software required to perform "vehicle, crew, or primary mission function," as defined by software that is:

1. Required to operate the vehicle or space asset (e.g., spacesuit, rover, or outpost), including commanding of the vehicle or asset, or

2. required to sustain a safe, habitable environment for the crew, or

3. required to achieve the primary mission objectives, or

4. directly prepares resources (e.g., data, fuel, power) that are consumed by the above functions.

* Includes software involving launch, onorbit, in space, surface operations, and entry, descent, and landing.

---

[1] Current standards that address habitability and environmental health, including atmospheric composition and pressure, air and water quality and monitoring, acceleration, acoustics, vibration, radiation, thermal environment, combined environmental effects, and human factors, are documented in NASA-STD-3000, Volumes I-II, Man-Systems Integration Standards.

Examples:

Examples of Class A software (human rated space flight) include but are not limited to:

During Launch: abort modes and selection; separation control; range safety; crew interface (display and controls); crew escape; critical systems monitoring and control; guidance, navigation, and control; and communication and tracking.

On Orbit/In Space: Extra Vehicular Activity (EVA); control of electrical power; payload control (including suppression of hazardous satellite and device commands); critical systems monitoring and control; guidance, navigation, and control; life support systems; crew escape; rendezvous and docking; failure detection; isolation and recovery; communication and tracking; and mission operations.

On Ground: pre-launch and launch operations; Mission Control Center (and Launch Control Center) front end processors; spacecraft commanding; vehicle processing operations; and re-entry operations; flight dynamics simulators used for ascent abort calls; and launch and flight controller stations for manned spaceflight.

Entry, Descent and Landing (EDL): command and control; aero-surface control; power; thermal; and fault protection; and communication and tracking.

Surface Operations: planet/lunar surface EVA; and communication and tracking.

Exclusions:

Class A does not include:

1. Software which happens to fly in space but is superfluous to mission objectives (e.g., software contained in an iPod carried on board by an astronaut for personal use), or

2. software that exclusively supports aeronautics, Research and Technology, and science conducted without spaceflight applications, or

3. systems (e.g., simulators, emulators, stimulators, facilities) used to test Class A systems containing software in a development environment.

Class B: Non-Human Space Rated Software Systems or Large Scale Aeronautics Vehicles

Definition:

Space Systems*: Flight and ground software that must perform reliably to accomplish primary mission objectives, or major function(s) in Non-Human Space Rated Systems. Limited to software that is:

1. Required to operate the vehicle or space asset (e.g., orbiter, lander, probe, flyby spacecraft, rover, launch vehicle, or primary instrument), such as commanding of the vehicle or asset, or

2. required to achieve the primary mission objectives, or

3. directly prepares resources (data, fuel, power, etc.) that are consumed by the above functions.

Airborne Vehicles: Large Scale1 aeronautic vehicles that are NASA unique in which the software:

1. Is integral to the control of an airborne vehicle, or

2. monitors and controls the cabin environment, or

3. monitors and controls the vehicle's emergency systems.

This definition includes software for vehicles classified as "test", "experimental", or "demonstration" which meets the above definition for Class B software. Also included are systems in a test or demonstration where the software's known and scheduled intended use is to be part of a Class A or B software system.

* Includes software involving launch, onorbit, in space, surface operations, and entry, descent, and landing.

---

[1] Large-scale (life-cycle cost exceeding $250M) fully integrated technology development system — see NPR 7120.8, section 5.1.1.1.

Examples:

Examples of Class B software includes but are not limited to:

Space, Launch, Ground, EDL, and Surface Systems: propulsion systems; power systems; guidance navigation and control; fault protection; thermal systems; command and control ground systems; planetary/lunar surface operations; hazard prevention; primary instruments; science sequencing engine; simulations which create operational EDL parameters; subsystems that could cause the loss of science return from multiple instruments; flight dynamics and related data; launch and flight controller stations for non-human spaceflight.

Aeronautics Vehicles (Large Scale NASA Unique): guidance, navigation, and control; flight management systems; autopilot; propulsion systems; power systems; emergency systems (e.g., fire suppression systems, emergency egress systems; emergency oxygen supply systems, traffic/ground collision avoidance system); and cabin pressure and temperature control.

Exclusions:

Class B does not include:

1. Software that exclusively supports non-primary instruments on Non-Human Space Rated Systems (e.g., low cost non-primary university supplied instruments) or

2. systems (e.g., simulators emulators, stimulators, facilities) used in testing Class B systems containing software in a development environment.

Class C: Mission Support Software or Aeronautic Vehicles, or Major Engineering/Research Facility Software

Definition:

Space Systems:

1. Flight or ground software that is necessary for the science return from a single (non-primary) instrument, or

2. flight or ground software that is used to analyze or process mission data, or

3. other software for which a defect could adversely impact attainment of some secondary mission objectives or cause operational problems, or

4. software used for the testing of space assets, or

5. software used to verify system requirements of space assets by analysis, or

6. software for space flight operations, that is not covered by Class A or B.

Airborne Vehicles:

Systems for non-large scale aeronautic vehicles in which the software:

1. is integral to the control of an airborne vehicle, or

2. monitors and controls the cabin environment, or

3. monitors and controls the vehicle's emergency system.

Systems on an airborne vehicle (including large scale vehicles) that acquire, store, or transmit the official record copy of flight or test data.

Major Engineering/Research Facility: Systems that operate a major facility for research, development, test, or evaluation (e.g., facility controls and monitoring, systems that operate facility-owned instruments, apparatus, and data acquisition equipment).

Examples: Examples of Class C software include but are not limited to:

Space Systems: software that supports prelaunch integration and test; mission data processing and analysis; analysis software used in trend analysis and calibration of flight engineering parameters; primary/major science data collection storage and distribution systems (e.g., Distributed Active Archive Centers); simulators, emulators, stimulators, or facilities used to test Class A, B, or C software in a development, integration and test environments (development environment includes environments used from unit testing through validation testing); software used to verify system-level requirements associated with Class A, B, or C software by analysis (e.g., guidance, navigation and controls (GN&C) system performance verification by analysis); simulators used for mission training; software employed by network operations and control (which is redundant with systems used at tracking complexes); command and control of non-primary instruments; ground mission support software used for secondary mission objectives, real-time analysis, and planning (e.g., monitoring, consumables analysis, mission planning).

Aeronautics Vehicles: guidance, navigation, and control; flight management systems; autopilot; propulsion systems; power systems; emergency systems (e.g., fire suppression systems, emergency egress systems, emergency oxygen supply systems, traffic/ground collision avoidance system); cabin pressure and temperature control; in-flight telescope control software; aviation data integration systems; and automated flight planning systems.

Major Engineering/Research Facility: major Center facilities; data acquisition and control systems for wind tunnels, vacuum chambers, and rocket engine test stands; ground-based software used to operate a major facility telescope; and major aeronautic applications facilities (e.g., air traffic management systems; high fidelity motion based simulators).

Exclusions: Systems unique to a research, development, test, or evaluation activity in a Major Engineering/Research Facility or Airborne Vehicle where the system is not part of the facility or vehicle and does not impact the operation of the facility or vehicle.

Class D: Basic Science/Engineering Design and Research and Technology Software

Definition:

Basic Science/Engineering Design:

1. Ground software that performs secondary science data analysis, or

2. ground software tools that support engineering development, or

3. ground software used in testing other Class D software systems, or

4. ground software tools that support mission planning or formulation, or

5. ground software that operates a research, development, test, or evaluation laboratory (i.e., not a Major Engineering/Research Facility), or

6. ground software that provides decision support for non-mission critical situations.

Airborne Vehicle Systems:

1. Software whose anomalous behavior would cause or contribute to a failure of system function resulting in a minor failure condition for the airborne vehicle (e.g., the Software Considerations in Airborne System and Equipment Certification, DO-178B, "Class D"), or

2. software whose anomalous behavior would cause or contribute to a failure of system function with no effect on airborne vehicle operational capability or pilot workload (e.g., the Software Considerations in Airborne System and Equipment Certification, DO-178B, "Class E"), or

3. ground software tools that perform research associated with airborne vehicles or systems.

Major Engineering/Research Facility Related: research software that executes in a Major Engineering/Research Facility but is independent of the operation of the facility.

Examples:

Examples of Class D software includes but are not limited to:

Basic Science and Engineering Design: engineering design and modeling tools (e.g., Computer-Aided Design and Computer-Aided Manufacturing (CAD/CAM), thermal/structural analysis tools); project assurance databases (e.g., problem reporting, analysis, and corrective action system, requirements management databases); propulsion integrated design tools; integrated build management systems; inventory management tools; probabilistic engineering analysis tools; test stand data analysis tools; test stand engineering support tools; experimental flight displays evaluated in a flight simulator; and tools used to develop design reference missions to support early mission planning.

Airborne Vehicles: software tools for designing advanced human-automation systems; experimental synthetic-vision display; and cloud-aerosol Light Detection and Ranging (LIDAR) installed on an aeronautics vehicle.

Exclusions:

Class D does not include:

1. Software that can impact primary or secondary mission objectives or cause loss of data that is

generated by space systems, or

2. software which operates a Major Engineering/Research Facility, or

3. software which operates an airborne vehicle, or

4. space flight software.

Class E: Small Light Weight Design Concept and Research and Technology Software

Definition:

1. Software developed to explore a design concept or hypothesis, but not used to make decisions for an operational Class A, B, or C system or to-be built Class A, B, or C system, or

2. software used to perform minor desktop analysis of science or experimental data.

Examples: Examples of Class E software include but are not limited to:

parametric models to estimate performance or other attributes of design concepts; software to explore correlations between data sets; line of code counters; file format converters; and document template builders.

Exclusions:

Class E does not include:

1. Space flight systems, or

2. software developed by or for NASA to directly support an operational system (e.g., human rated space system, robotics spacecraft, space instrument, airborne vehicle, major engineering/research facility, mission support facility, primary/major science data collection storage and distribution systems), or

3. software developed by or for NASA to be flight qualified to support an operational system, or

4. software that directly affects primary or secondary mission objectives, or

5. software that can adversely affect the integrity of engineering/scientific artifacts, or

6. software used in technical decisions concerning operational systems, or

7. software that has an impact on operational vehicles.

Business and IT Infrastructure Software

Class F: General Purpose Computing Software (Multi-Center or Multi- Program/Project)

Definition: General purpose computing software used in support of the Agency, multiple Centers, or multiple programs/projects, as described for the General Purpose Infrastructure To-Be Component of the NASA Enterprise Architecture, Volume 5 (To-Be Architecture), and for the following portfolios: voice, wide area network, local area network, video, data Centers, application services, messaging

and collaboration, and public Web. A defect in Class F software is likely to affect the productivity of multiple users across several geographic locations and may possibly affect mission objectives or system safety. Mission objectives can be cost, schedule, or technical objectives for any work that the Agency performs.

Examples: Examples of Class F software include but are not limited to: software in support of the NASA-wide area network; the NASA Web portal; and applications supporting the Agency's Integrated Enterprise Management Program, such as the time and attendance system, Travel Manager, Business Warehouse, and E-Payroll.

Class G: General Purpose Computing Software (Single Center or Project)

Definition: General purpose computing software used in support of a single Center or project, as described for locally deployed portions of the General Purpose Infrastructure To-Be Component of the NASA Enterprise Architecture, Volume 5 (To-Be Architecture) and for the following portfolios: voice, local area network, video, data Centers, application services, messaging and collaboration, and public Web. A defect in Class G software is likely to affect the productivity of multiple users in a single geographic location or workgroup but is unlikely to affect mission objectives or system safety.

Examples: Examples of Class G software include but are not limited to: software for Center custom applications such as Headquarters' Corrective Action Tracking System and Headquarters' User Request Systems.

Class H: General Purpose Desktop Software

Definition: General purpose desktop software as described for the General Purpose Infrastructure To-Be Component (Desktop Hardware and Software Portfolio) of the NASA Enterprise Architecture, Volume 5 (NASA To-Be Architecture). A defect in Class H software may affect the productivity of a single user or small group of users but generally will not affect mission objectives or system safety. However, a defect in desktop IT security-related software, e.g., anti-virus software, may lead to loss of functionality and productivity across multiple users and systems.

Examples: Examples of Class H software include but are not limited to: desktop applications such as word processing applications, spreadsheet applications, and presentation applications.