NASA Procedural Requirements

NPR 7150.2
Effective Date: September 27, 2004
Expiration Date: September 27, 2009

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) 2820.1, NASA Software Policies. This NPR provides the minimal set of requirements established by the Agency for software acquisition, development, maintenance, 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 thorough, but not all inclusive, set of software engineering requirements in generic terms to be applied throughout NASA and its contractor community.

P.2 Applicability and Scope

P.2.1 The requirements of this NPR cover software created or acquired by or for NASA, including commercial-off-the-shelf software (COTS), government-off-the-shelf software (GOTS), modified-off-the-shelf software (MOTS), open source, reuse, legacy, and heritage software. Requirements in this NPR apply to all of the Agency's product lines containing software systems and subsystems. The applicability of requirements in this NPR to specific systems and subsystems within Agency product lines, programs, and projects is determined through the use of the NASA-wide definition of software classes in Appendix B, in conjunction with the Requirements Mapping Matrix in Appendix D. It is not uncommon for a project to contain multiple systems and subsystems having different software classes. Through the use of the Requirements Mapping Matrix, the number of applicable requirements and their associated rigor are scaled back for less critical software classes.

P.2.2 This NPR applies to NASA Headquarters, NASA Centers, and NASA Component Facilities.

P.2.3 This NPR shall be applied to software development, maintenance, operations, management, acquisition, and assurance activities started after its effective date of issuance [SWE-001].

Note: This document is not applicable to software development, maintenance, operations, management, acquisition, and assurance activities started before its effective date of issuance (i.e., existing systems and subsystems containing software for Shuttle, International Space Station, Hubble, Chandra, etc.). If the respective Governing Program Management Council (GPMC) determines that an existing software activity should follow all or part of this NPR, the results of that decision should be documented in the Project Plan (as defined in NPR 7120.5, NASA Program and Project Management Processes and Requirements). The respective GPMC can make this determination based on the safety criticality of the existing project, the mission criticality, project cost, current phase of the existing program, etc.

P.2.4 This NPR provides procedural requirements to the responsible NASA project managers and contracting officers for NASA contracts. It 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. NASA program and project management requirements are contained in NPR 7120.5, NASA Program and Project Management Processes and Requirements. 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 responsible party for each requirement is identified by an underline.

P.3 Authority

a. 29 U.S.C. 749d, Section 508 of the Rehabilitation Act of 1973, as amended. Specific requirements for accessibility may be found at 36 CFR Part 1194, available at http://www.access-board.gov/sec508/508standards.htm.

b. 35 U.S.C. 200 Chapter 18 - Patent Rights in Inventions Made with Federal Assistance.

c. 40 U.S.C. 1401, et seq. Section 808 of Public Law 104-208, the Clinger-Cohen Act of 1996 [renaming, in pertinent part, the Information Technology Management Reform Act (ITMRA), Division E of Public Law 104-106].

d. 42 U.S.C. 2457 Property Rights in Inventions.

e. 42 U.S.C. 2473(c)(1) of the National Aeronautics and Space Act of 1958, as amended.

f. 44 U.S.C 3501 et seq., Paperwork Reduction Act of 1994 (Public Law 104-13).

g. OMB Circular A-130, Management of Federal Information Resources.

P.4 References

See Appendix A for a complete reference list of all documents cited in this NPR.

P.5 Cancellation

None.

/S/
Theron M. Bradley, Jr.
NASA Chief Engineer

CHAPTER 1: Introduction

1.1 Overview

1.1.1 This NPR provides a common set of generic requirements for software created and acquired by or for NASA. Software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software: that is, the application of engineering to software. This NPR is designed to be a stand-alone compendium 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 are easily mapped to industry standards and proven NASA experience in software engineering. Centers and software developers will find that many of the requirements are already satisfied through programs, procedures, and processes that are already in place.

1.1.3 The Agency will make significant investments in software engineering to support the Agency's product lines: Flight Products, Advanced Technology Development, Sustaining Operations, Functional Infrastructure, and Basic and Applied Research. NASA must ensure that programs, projects, systems, and subsystems that utilize software follow a basic set of requirements. A One NASA approach is being used 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 life cycle model. Each has its strengths and weaknesses, and no one model is best for all situations. It is important to evaluate the potential life cycle models and select one that best matches the product you are producing. Standards or organizational policy may dictate a particular life cycle model.

1.1.5 The 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 an updated version 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 Independent Technical Authority (ITA) warrant authorities.

1.2 Organizational Capability 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.

1.2.1 The NASA Chief Engineer shall lead, maintain, and fund an Agencywide 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 requirements for the content of each Center Software Engineering Improvement Plan are defined in Chapter 5. Each Center has a current Center Software Engineering Improvement Plan on file in the Office of the Chief Engineer.

1.2.3 The NASA 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 should be documented per Center Software Engineering Improvement Plan requirements. Capability Maturity Model® Integration (CMMI®) - systems engineering and software engineering (CMMI®-SE/SW) appraisals are the preferred benchmark 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 Chief Engineer, in coordination with the Chief Safety and Mission Assurance Officer, shall maintain a process that provides, on a recurring basis, 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 text that follows the figure provides a brief description of each type of document, listed according to its position in the

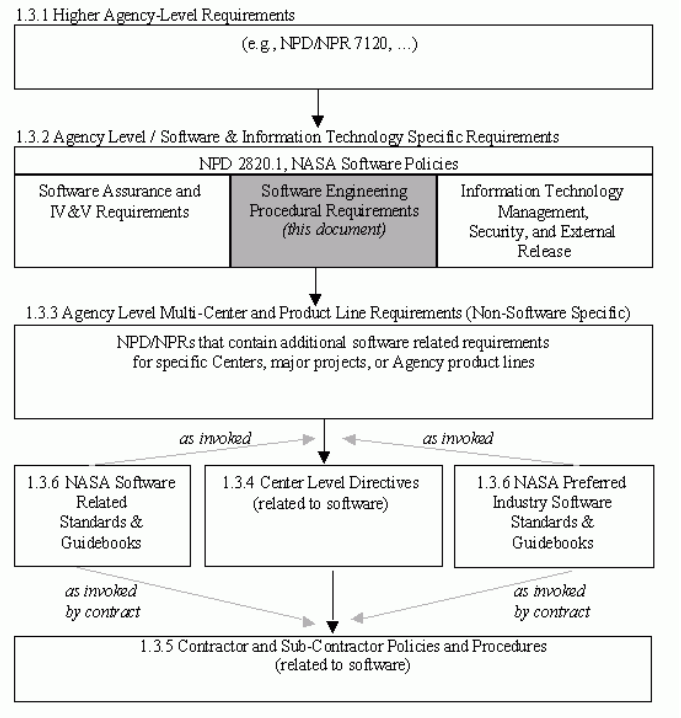

figure.

FIGURE 1-1 Relationships Among Governing Software Documents

1.3.1 Higher Agency-Level Requirements
These NPDs and NPRs document the overarching management system for the Agency and describe requirements that cut across several disciplines. Examples of relevant higher level documents of this nature include: NPD 1000.1, NASA Strategic Plan; NPR 1000.2, NASA Strategic Management Handbook; NPR 1000.3, The NASA Organization; NPD 7120.4, Program/Project Management; and NPR 7120.5, NASA Program and Project Management Processes and Requirements. 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 and Information Technology Specific Requirements
NPD 2820.1, NASA Software Policies, is an overarching document that establishes top-level policies for all software created and acquired by or for NASA. This NPR elaborates and provides further detail on the requirements in NPD 2820.1 within the discipline of engineering during software acquisition, development, maintenance, operations, and management. Additional requirements in closely related areas have been established by the Office of Safety and Mission Assurance, the Chief Information Officer, Exploration Systems Mission Directorate, and Office of Security and Program Protection.

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.
An example of a representative NPR in this category is NPR 8705.2, Human-Rating Requirements for Space Flight Systems.

1.3.4 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.

1.3.5 Contractor and Subcontractor Policies and Procedures
Contractors and subcontractors develop in-house policies and procedures to provide quality 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 a number of different customers in an effective and efficient manner.

1.3.6 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.

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 other NASA policy requirements that impact the development, release, and/or maintenance of the software.

2.1.1 The project shall ensure that software disclosure requirements of NPD 2091.1, Inventions Made By Government Employees, are implemented by their project, Section 305 of the Space Act (42 U.S.C 2457) for large business contractors, and 35 U.S.C. 200 et seq., (including Section 202(c)) for small businesses, universities, and non-profits are implemented by their project. [SWE-007]

2.1.2 The project shall ensure that software technology transfer requirements of NPR 2190.1, NASA Export Control Program, are implemented by the project. [SWE-008]

2.1.3 The project shall ensure that software external release requirements of NPR 2210.1, External Release of NASA Software, are implemented by the project. [SWE-009]

2.1.4 The project shall ensure that the security requirements of NPD 2810.1, NASA Information Security Policy, are implemented by the project. [SWE-010]

2.1.5 The project shall ensure that the requirements of reasonable accommodation for individuals with disabilities per NPR 3713.1, Procedures for Providing Reasonable Accommodation for Individuals with Disabilities, are implemented by the project. [SWE-011]

2.1.6 The project shall ensure that software is accessible to individuals with disabilities as required by Section 508 of the Rehabilitation Act (29 U.S.C. 749d), as amended. Specific requirements for accessibility may be found at36 CFR Part 1194. [SWE-012]

2.2 Software Life Cycle Planning

Software Life Cycle Planning covers the software aspects of a project from inception through retirement. It is meant as 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 greater project. At project conception, software needs for the project are analyzed, including acquisition, supply, development, operation, maintenance, 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 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.

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

2.2.3 The project shall establish, document, and maintain at least one software cost estimate 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 an assessment of the technology to be used and the impact on risk, cost, and schedule.

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.

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

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 The project shall classify each of the systems and subsystems containing software in accordance with the software classifications definitions for Class A, B, C, D, E, F, G and H in Appendix B. [SWE-020]

Note: These classifications are documented in the Software Development or Management Plan as defined in Chapter 5. Software Assurance also performs an independent classification assessment and the results should be compared as per NASA-STD-8739.8, NASA Software Assurance.

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

2.2.10 The project shall ensure that software assurance is implemented by their project as per NASA-STD-8739.8, NASA Software Assurance. [SWE-022]
Note: Software assurance activities occur throughout the life of the project and, while some of the actual analyses may be performed within 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). It is important to have software assurance of all software activities and products including Request for Proposals, contracts and memorandums of agreement, software plans, requirements, design, implementation, verification, validation, certification, acceptance, maintenance, operations, and retirement.

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, are implemented by the project. [SWE-023]

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

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

2.2.14 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, and Modified Off-The-Shelf Software

Since many of NASA's projects now contain off-the-shelf software products, it is important to plan and manage when and how to incorporate them. The off-the-shelf software discussed here apply only when those off-the-shelf software are to be included as part of a NASA system (per section P.2.1). The following requirements do not apply to standalone desktop

applications (e.g. MS Word, Adobe Acrobat, MS Power Point). However, when applications such as MS Excel or MS Access are used within a NASA system/subsystem as defined by the classes in this NPR, then they will need to be assessed and classified as part of the software subsystem in which they reside.

2.3.1 The project shall ensure that when COTS, GOTS, MOTS, open source, reuse, legacy, or heritage software product is to be acquired, the following conditions are satisfied: [SWE-027]

a. The requirements that are to be met by the off-the-shelf software are identified.

b. The off-the-shelf software includes documentation to fulfill its intended purpose (e.g. usage instructions).

c. Proprietary, usage, ownership, warranty, licensing rights, and transfer are addressed.

d. Future support for the off-the-shelf software product is planned.

e. Off-the-shelf software is validated to the same level of confidence as would be required of the developed software.

Note: It is the responsibility of the organization proposing to procure off-the-shelf software to document, prior to procurement, the plan for validating that such software can be assigned the same level of confidence that would be needed for an equivalent class of software if obtained through a "development" process.

Note: For critical systems or systems which must be maintained for long periods of time beyond the time a supplier would maintain or support the software the following should be considered:

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 join a product users group to obtain this information.

d. The plan to provide adequate support is in place, including timely maintenance and cost of maintenance.

e. Any changes to the software management, development, operations, or maintenance plans that are affected by the use or incorporation of COTS, GOTS, MOTS, reuse, legacy, or heritage software should be documented by the project.

2.4 Software Verification and Validation

Ensuring that the software products meet their requirements 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, peer reviews/inspections, testing, demonstration, and analyses all 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 peer reviews are such an important verification tool with proven value, there are specific peer review requirements 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 demonstrates the software products meet specified requirements. Methods of software verification include: 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, analyses of requirement implementation, and software product demonstrations. Planning for software verification should address the development, management review, and documentation for the software products. Refer to the Software Development or Management Plan software documentation requirement for software verification planning and incorporation (Chapter 5).

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 demonstrates the as-built software product or software product component satisfies its intended use in its intended environment. Methods of software validation include: peer reviews/inspections of software product component behavior in a simulated environment, acceptance testing against mathematical models, analyses, and operational environment demonstrations. Planning for software validation should address the development, maintenance, support, and training for the software product and software product components. Refer to the Software Development or Management Plan software documentation requirement 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. While in the past, software planning was often left until later in the project life cycle, it is now seen as 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 and schedule estimates, work planning, Request for Proposals, the evaluation of the contractors, and the contracts themselves.

2.5.1 Consistent with the Requirements Mapping Matrix (Appendix D), the project shall ensure that software is developed by either a software CMM® Maturity Level 3 or higher organization; or by an organization that has a CMMI®-SE/SW Capability Level 2 or higher as measured by a Software Engineering Institute (SEI) authorized lead appraiser from an external organization in the following Process Areas: [SWE-032]

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.

Note: Organizations who have completed Standard CMMI® Appraisal Method for Process Improvement (SCAMPI) Class A appraisals against the CMMI® Model are expected to have their results posted on the SEI web site so that NASA can assess the current maturity state in the selection process.

2.5.2 The project shall assess options for acquisition against analysis of appropriate criteria to include risk, cost, and benefits for each option listed below: [SWE-033]

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. A combination of a, b, and c above.

e. Enhance an existing software product or service.

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 appropriate 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 monitoring of the acquisition. [SWE-037]

Note: All 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 ultimate acquisition success. This section contains the software acquisition requirements that should be performed by NASA organizations acquiring systems and/or services.

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, including monitoring integration and verification adequacy, trade study data, auditing the software development process, and participation in all software reviews and technical interchange meetings. [SWE-039]

2.6.1.2 The project shall require the software supplier(s) to provide NASA all software products and software process tracking information, in electronic format, including all 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 Request for Proposals, 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 modified off-the-shelf software and non-flight software (ground test software, simulations, ground analysis software, ground control software, science data processing software, hardware manufacturing software, or other). [SWE-042]

Note: All known contract requirements should be addressed in the Request for Proposals 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 should consider having the software copyright assigned to the government. This should be addressed in the Request for Proposals. The project should consult the Center Chief of Patent/Intellectual Property Counsel regarding required rights in the software.

2.6.2 Supplier Monitoring Requirements

2.6.2.1 The project shall require the software supplier to track all software changes 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 requirements for the content of the Software Metric Report are 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 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 life cycle model. Each has its strengths and weaknesses, and no one model is best for all situations. 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. 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. It is important that there is ongoing customer validation of the requirements to ensure the end products meet the customer needs. 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 each Software Requirement Specification and Data Dictionary are defined in Chapter 5.

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: This analysis is for safety criticality, correctness, consistency, clarity, completeness, traceability, feasibility, verifiability, and maintainability. This includes the allocation of functional and performance requirements to functions and subfunctions.

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

Note: The project should identify any orphaned or widowed requirements (no parent or no child) associated with reused software.

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 should analyze and document changes to requirements for cost, technical, and schedule impacts.

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

Note: A verification matrix supports the accomplishment of this requirement.

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: This should include confirmation that the requirements meet the needs and expectations of the customer.

3.2 Software Design

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 the software design. [SWE-056]

Note: The requirements for the content of the software design description and interface design description are defined in Chapter 5.

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

3.2.3 The project shall develop and record a detailed design based on the 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 bi-directional 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 the software code is unit tested per the plans for software testing. [SWE-062]

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

Note: The requirements for the content of the Software Version Description document are defined in Chapter 5.

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

3.4 Software Testing

The purpose of testing is to verify the software 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 should also validate that the software operates appropriately in the intended environment.

3.4.1 The project shall provide: [SWE-065]

a. Software Test Plan(s).

b. Software Test Procedures.

c. Software Test Reports.

Note: The requirements for the content of the Software Test Plan, Software Test Procedures, and Software Test Reports are defined in Chapter 5.

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

Note: Testing could include software integration testing, systems integration 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. Automated testing tools should also be considered.

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 test, validate, and certify software models, simulations, and analysis tools. [SWE-070]

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 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]

3.5 Software Operations, Maintenance, and Retirement

Planning for operations, maintenance, and retirement begins early in 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 it is delivered to the customer until retirement.

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

Note: The requirements for the content of the Software Maintenance Plan are 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 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 should be reviewed by the Center Chief of Patent/Intellectual Property Counsel before being accepted into software development projects. Other documentation that should be considered for delivery is:

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 Discrepancy 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 typically do not happen in one 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, product) 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 establishes and maintains the integrity of the products of a software project throughout the software life cycle. Software configuration management involves identifying the configuration of products that are delivered to the customer and used in development, systematically controlling changes to the configuration, and maintaining the integrity and traceability of the configuration. 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 plan may be a part of the project configuration management plan with required content of the plan defined 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 or software problem tracking system.
Software Change Requests/Problem Reports should be documented per the requirements in Chapter 5.

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

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 provides 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 (potential issues, hazards, threats, and vulnerabilities) in accordance with NPR 7120.5, NASA Program and Project Management Processes and Requirements and NPR 8000.4, Risk Management Procedural Requirements. [SWE-086]

4.3 Peer Reviews/Inspections

Peer reviews and inspections are the in-process technical examination of work products by the supplier's peers for the purpose of finding and eliminating defects early in the life cycle. Peer reviews 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. Peer reviews and inspections which satisfy these features include, but are not limited to, Fagan Inspections, Software Formal Inspections, Tom Gilb's Software Inspections, and Perspective Based Reading.

4.3.1 The project shall ensure peer reviews are performed for: [SWE-087]

a. Software Requirements.

b. Software Test Plans.

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

Note: Safety and mission-success related design and code components should be peer reviewed.

4.3.2 The project shall, for each planned peer review: [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.

4.3.3 The project shall, for each planned peer review, record basic measurements. [SWE-089]

Note: The requirements for the content of the Software Inspection/Peer Review Report are defined in Chapter 5.

4.4 Software Measurement

Software measurement programs at multiple levels should be established to meet measurement objectives that are derived from identified information needs and 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-level (product line) measurement programs should be 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, 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: Metrics reports should be documented per the metrics report requirements of 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]

Note: Data should be maintained in the NASA process asset library.

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]

4.4.8 Each NASA Mission Directorate shall report their software measurement results to the Office of the Chief Engineer on a yearly basis. [SWE-097]

4.5 Best Practices

Successful best practices throughout the software community provide an available resource that can lead to improved 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.

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

Note: The repository may contain information in many forms including, but not limited to, websites, design principles, books, periodicals, presentations, 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

Properly trained personnel are key to success with 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 should include 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 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 a 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 should be documented per the Software Training Plan requirements of Chapter 5. Centers should plan to meet or exceed the CMMI® SE/SW Maturity Level 3.

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 all projects. Non-applicable content areas will be clearly noted in project documentation. These non-applicable documentation decisions may be reviewed by external organizations. Product documentation should be 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 (i.e., Safety and Mission Assurance, Independent Verification and Validation (IV&V), Independent Technical Authority (ITA), NASA Engineering and Safety Center (NESC)).

b. The classification of each of the systems and subsystems containing software as defined in Appendix B.

c. Tailoring compliance matrix for approval by the designated ITA Warrant Authority, if the projects has any variants, waivers or exceptions 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, nondeliverable items to be performed, budgets, staffing, 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 approach.

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. 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.

Note: Verification approach includes:

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

b. Identification of selected work products to be verified (e.g., peer reviews of requirements and test plans, peer reviews/inspections of critical code, testing code against requirements and design).

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, peer review records, inspection records), and where software verification corrective action will be documented.

Note: Validation approach includes:

a. Identification of selected software validation procedures 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 (e.g., user groups reviewing requirements and prototypes, acceptance testing of software product, operational demonstrations of software product).

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, 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) within which Software Configuration Management is to apply.

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.

b. Test types (e.g., unit testing, software integration testing, systems integration testing, end-to-end testing, acceptance testing, regression testing).

c. Test classes.

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.

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 labs required for software verification and implementation.

(4) Updates to documentation for modified COTS or non-COTS software.

(5) Licensing agreements for COTS.

(6) Plan for and tracking of operational backup software.

(7) Approach for the implementation of modifications to operational software (e.g., testing of software in development lab 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, document, and gain concurrence with the Office of Safety and Mission Assurance as to the extent and responsibilities of the assurance and safety of the project. This will be documented into the project plans and reflected in the assurance process.

5.1.5.1 The Software Assurance Plan(s) shall be written 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.

5.2 Software Requirements and Product Data

5.2.1 Software Requirements Specification
The Software Requirements Specification details the software performance, interface, operational, and quality assurance requirements for each 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.

(7) Safety requirements.

(8) Performance and timing requirements.

(9) Security and privacy requirements.

(10) Environment requirements.

(11) Computer resource requirements.

(a) Computer hardware resource 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.

d. Requirements traceability and verification data.

e. Requirements partitioning for phased delivery.

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

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, Multiplexer-Demultiplexer hardware channelization).

b. 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., on-board, ground, test specific).

h. Effector 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 input/output description.

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

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

(e) Requirements 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.

d. CSCI Implementation Plan.

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 (i.e., real-time data transfer, storage-and- retrieval of data) to be implemented.

c. Specification of individual data elements, 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 data element assemblies, format, and data content including bit-level descriptions of data interface 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 that affects an interface.

i. Interface compatibility.

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 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 change, including impact to safety, schedules, cost, products, and 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.

Note: The Software Change Request/Problem Report provides a means for identifying and recording the resolution to software anomalous behavior, process noncompliance 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 criteria for evaluating results, and 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 and overview of operation, contingencies and alternate states and modes of operation, security and privacy, and 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 backup, recovery from errors, malfunctions, emergencies, and messages.

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

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 software versions.

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

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

b. Executable software (i.e., 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).

5.3 Software Report Requirements

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 product 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, successfully 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).

c. Computer resource utilization in percentage of capacity.

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 peer reviews/software inspections (planned vs. actual).

d. Peer review 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.

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 product 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 (new, slightly modified, COTS) - planned vs. actual.

To the extent information regarding 5.3.1.a through 5.3.1.e of SWE-117 is not provided, the project will provide documented justification in the Software Metrics Report. 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 assessment of the software as demonstrated 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 Inspection/Peer Review Report

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

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

b. Summary on total time expended on each inspection/peer review (including total hour summary and time participants spent reviewing 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, etc.)).

e. Inspection results summary (i.e., pass, re-inspection required).

f. Listing of all inspection defects.

CHAPTER 6: Tailoring, Warrant 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 and independently maintain control over the application of requirements in this NPR and to ensure proposed variants from specific requirements are appropriately mitigated, the Office of the Chief Engineer has established a "Warrant Authority" within NASA's Independent Technical Authority (ITA). Variants from requirements in this NPR are governed by the following requirements.

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 designated Center-level requirements approved by the authorized Center ITA Manager.

6.1.1 For those cases in which a Center, project, or program desires to apply specific or general practices that do not meet or exceed the requirements of this NPR, the Center shall recommend those alternate requirements for Agency Technical Authority approval with appropriate justification and Center ITA Manager concurrence. [SWE-120]

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

6.2 Expertise of ITA Warrant Authority(s)

6.2.1 The designated ITA Warrant Authorities for this NPR for non- business and non-IT infrastructure systems shall be approved by the NASA Chief Engineer, in coordination with the NASA Chief Safety and Mission Assurance Officer. [SWE-122]

Note: The designated ITA Warrant Authorities for this NPR should be recognized NASA software engineering experts.

6.2.2 The designated ITA Warrant Authorities for this NPR for business and IT-infrastructure systems shall be approved by the NASA Chief Information Officer. [SWE-123]

6.3 Compliance

6.3.1 The designated ITA Warrant Authority for this NPR shall comply with Office of the Chief Engineer's direction for NASA Independent Technical Authority. [SWE-124]

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]

6.3.3 The designated ITA Warrant Authority for this NPR shall consider the following information when assessing waivers and variants from requirements in this NPR: [SWE-126]

a. The list of Agency projects containing software.

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

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.

6.3.4 The designated ITA Warrant Authority for this NPR shall review and have concurrence approval for Center defined subsets of requirements denoted by "P(Center)" in the Requirements Mapping Matrix in Appendix D for the indicated Classes of software. [SWE-127]

6.3.5 The designated ITA Warrant Authority shall keep records of projects and organizational compliances, waivers, variants, and exceptions against this NPR; and submit an annual report to the Office of Chief Engineer, Office of Safety and Mission Assurance, and the Chief Information Officer. [SWE-128]

6.3.6 The Office of the Chief Engineer shall authorize appraisals against selected requirements in this NPR (or ITA approved alternative set of designated Center requirements) to check compliance. [SWE-129]

APPENDIX A: References

A.1 Applicable References

a. 29 U.S.C. 794d, Section 508 of the Rehabilitation Act of 1973, as amended. Specific requirements for accessibility may be found at 36 CFR Part 1194.

b. 40 U.S.C. 1401, et seq. Section 808 of Public Law 104-208, the Clinger-Cohen Act of 1996 [renaming, in pertinent part, the Information Technology Management Reform Act (ITMRA)], Division E of Public Law 104-106.

c. 42 U.S.C. 2473(c)(1) of the National Aeronautics and Space Act of 1958, as amended.

d. 44 U.S.C 3501 et seq., Paperwork Reduction Act of 1994 (Public Law 104-13).

e. OMB Circular A-130, Management of Federal Information Resources.

f. NPD 1000.1, NASA Strategic Plan.

g. NPD 2091.1, Inventions Made By Government Employees.

h. NPD 2210.1, External Release of NASA Software.

i. NPD 2810.1, NASA Information Security Policy.

j. NPD 2820.1, NASA Software Policies.

k. NPD 7120.4, Program/Project Management.

l. NPR 1000.2, NASA Strategic Management Handbook.

m. NPR 1000.3, The NASA Organization.

n. NPR 2190.1, NASA Export Control Program.

o. NPR 2210.1, External Release of NASA Software.

p. NPR 3713.1, Procedures for Providing Reasonable Accommodation for Individuals with Disabilities.

q. NPR 7120.5, NASA Program and Project Management Processes and Requirements.

r. NPR 8000.4, Risk Management Procedural Requirements.

s. NPR 8705.2, Human-Rating Requirements for Space Flight Systems.

t. NPR 8715.3, NASA Safety Manual.

u. NPR 8735.2, Management of Government Safety and Mission Assurance Surveillance Functions for NASA Contracts.

v. NASA-STD-8739.8, NASA Software Assurance Standard.

w. NASA-STD-8719.13, Software Safety Standard.

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

y. NASA Software Engineering website.

z. NASA Enterprise Architecture: Volume 5, NASA To-Be Architecture, Approach to Design and Implementation.

aa. CMU/SEI-2002-TR-011, CMMI¨a for Systems Engineering/Software Engineering/Integrated Product and Process Development/Supplier Sourcing, Version 1.1, Continuous Representation.

bb. CMU/SEI-2002-TR-012 - CMMI¨a for Systems Engineering/Software Engineering/Integrated Product and Process Development/Supplier Sourcing, Version 1.1, Staged Representation.

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

A.2 Related References

a. IEEE 830, Recommended Practices for Software Requirements Specifications.

b. IEEE 1059, Guide for Software Verification and Validation Plans.

c. IEEE/EIA 12207.0, Standard for Information Technology-Software Life Cycle Processes.

d. IEEE/EIA 12207.1, Standard for Information Technology-Software Life Cycle Processes-Life Cycle Data.

e. IEEE/EIA 12207.2, Standard for Information Technology-Software Life Cycle Processes-Implementation Considerations.

f. CMU/SEI-93-TR-25, Key Practices of the Capability Maturity Model.

g. Continuous Risk Management Guidebook, NTIS#:AD-A319533KKG, DTIC#:AD-A319533\6\XAB.

APPENDIX B: Definitions

Class A Human Rated Software Systems	Applies to all space flight software subsystems (ground and flight) developed and/or operated by or for NASA to support human activity in space and that interact with NASA human space flight systems. Space flight system design and associated risks to humans are evaluated over the program's life cycle, including design, development, fabrication, processing, maintenance, launch, recovery, and final disposal. Examples of Class A software for human rated space flight include but are not limited to: guidance; navigation and control; life support systems; crew escape; automated rendezvous and docking; failure detection, isolation and recovery; and mission operations.
Class B Non-Human Space Rated Software Systems	Flight and ground software that must perform reliably in order to accomplish primary mission objectives. Examples of Class B software for non-human (robotic) spaceflight include, but are not limited to, propulsion systems; power systems; guidance navigation and control; fault protection; thermal systems; command and control ground systems; planetary surface operations; hazard prevention; primary instruments; or other subsystems that could cause the loss of science return from multiple instruments.
Class C Mission Support Software	Flight or ground software that is necessary for the science return from a single (non-critical) instrument or is used to analyze or process mission data or other software for which a defect could adversely impact attainment of some secondary mission objectives or cause operational problems for which potential work-arounds exist. Examples of Class C software include, but are not limited to, 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 and distribution systems, major Center facilities, data acquisition and control systems, aeronautic applications, or software employed by network operations and control (which is redundant with systems used at tracking complexes). Class C software must be developed carefully, but validation and verification effort is generally less intensive than for Class B.
Class D Analysis and Distribution Software	Non-space flight software. Software developed to perform science data collection, storage, and distribution; or perform engineering and hardware data analysis. A defect in Class D software may cause rework but has no direct impact on mission objectives or system safety. Examples of Class D software include, but are not limited to, software tools; analysis tools, and science data collection and distribution systems.
Class E Development Support Software	Non-space flight software. Software developed to explore a design concept; or support software or hardware development functions such as requirements management, design, test and integration, configuration management, documentation, or perform science analysis. A defect in Class E software may cause rework but has no direct impact on mission objectives or system safety. Examples of Class E software include, but are not limited to, earth science modeling, information only websites (non- business/information technology); science data analysis; and low technical readiness level research software.
Class F General Purpose Computing Software (Multi-Center or Multi-Program/Project)	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 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 Financial 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)	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 of Class G software include, but are not limited to, software for Center custom applications such as Headquarters' Corrective Action Tracking System and Headquarters' ODIN New User Request System.
Class H: General Purpose Desktop Software	General purpose desktop software as described for the General Purpose Infrastructure To-Be Component (Desktop Hardware & Software Portfolio) of the NASA Enterprise Architecture, Volume 5 (NASA To-Be Architecture). This class includes software for Wintel, Mac, and Unix desktops as well as laptops. 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 of Class H software include, but are not limited to, desktop applications such as Microsoft Word, Excel, and Power Point, and Adobe Acrobat.
Contracted Software	Software created for a project by a contractor or subcontractor. Process requirements and safety analyses may be included. This is custom-made software, but not in-house.
Commercial-Off-The-Shelf (COTS) Software	Operating systems, libraries, applications, and other software purchased from a commercial vendor. Not customized for a particular project. Access to source code and documentation are often limited.
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 either to prevent failures.

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 all 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.)
Heritage	See legacy. See software reuse.
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 Assurance Surveillance Functions for NASA Contracts.)
Legacy	These are usually 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.
Mission Critical	Item or function that must retain its operational capability to assure mission success. (Definition from source document: NPR 8715.3, NASA Safety Manual.)
Modified Off-The-Shelf (MOTS) Software	When COTS, legacy, reuse, or heritage software is changed to a certain degree, usually more than 10%, then it 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;" however, 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.
Off-The-Shelf Software	Software not developed in-house or by a contractor for the specific project now underway. The software is general purpose or developed for a different purpose from the current project.
Oversight	Surveillance mode that is in line with the supplier's processes. The customer retains and exercises the right to concur or nonconcur with the supplier's decisions. Nonconcurrence must be resolved before the supplier can proceed. Oversight is a continuum that can range from low intensity, such as customer concurrence in reviews (e.g., PDR, CDR), to high intensity oversight, in which the customer has day-to-day involvement in the supplier's decision-making process (e.g., hardware inspections). (Definition from source document: NPR 8735.2, Management of Government Safety and Mission Assurance Surveillance Functions for NASA Contracts.)
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.)
Program	The term "program" is as defined in NPR 7120.5, NASA Program and Project Management Processes and Requirements.
Project	The term "project" is as defined in NPR 7120.5, NASA Program and Project Management Processes and Requirements.
Reuse	See software reuse.
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 Safety and Mission Assurance Surveillance Functions for NASA Contracts.)
Safety Critical Function	The term "safety critical function" is as defined in NPR 8715.3, NASA Safety Manual.
Safety Critical	The term "safety critical" is as defined in NPR 8715.3, NASA Safety Manual.
Software	Computer programs, procedures, rules, and associated documentation and data pertaining to the development and operation of a computer system. Software includes programs and operational data . This also includes COTS, GOTS, MOTS, reuse, auto code generated, firmware, and open source software components.
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.)
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 it is supplied from one government project to another government project. (Definition from source document: NASA-GB-8719.13, NASA Software Safety Guidebook.)

Surveillance	The continual monitoring and verification of status of an entity and analysis of records to ensure that specified requirements are being met. Note: Surveillance can be performed in an insight, oversight, or a combined mode as determined by NASA using a risk-based decision process. (Definition from source document: NPR 8735.2, Management of Government Safety and Mission Assurance Surveillance Functions for NASA Contracts.)
System	The combination of elements that function together to produce the capability required to meet a need. The elements include all hardware, software, equipment, facilities, personnel, processes, and procedures needed for this purpose. (Definition from source document: NPR 7120.5, NASA Program and Project Management Processes and Requirements.)
Validation	Proof that the product accomplishes the intended purpose. May be determined by a combination of test, analysis, and demonstration. (Definition from source document: NPR 7120.5, NASA Program and Project Management Processes and Requirements.) Note: Software validation also includes software peer review and inspection.
Verification	Proof of compliance with specifications. May be determined by a combination of test, analysis, demonstration, and inspection. (Definitions from source document: NPR 7120.5, NASA Program and Project Management Processes and Requirements.)
Wrappers	See glueware.

APPENDIX C: Acronyms

EMB	Engineering Management Board
FDIR	Failure Detection, Isolation, and Recovery
EPG	Engineering Process Group
ISO	International Organization for Standardization of Geneva, Switzerland
PAL	Process Asset Library
CDR	Critical Design Review
CIO	Chief Information Office
CMM®	Capability Maturity Model®
CMMI®	Capability Maturity Model® Integration
CMMI®-SE/SW	Capability Maturity Model® Integration for Systems Engineering and Software Engineering
CMU	Carnegie Mellon University
COTS	Commercial-Off-The-Shelf
CSCI	Computer Software Configuration Item
EIA	Electronic Industries Alliance; subsidiary of Government Electronics and Information Technology Association of Arlington, VA
FAR	Federal Acquisition Regulation
GOTS	Government-Off-The-Shelf
GPMC	Governing Program Management Council
HWCI	Hardware Configuration Item
I/O	Input/Output
IEEE	Institute of Electrical and Electronics Engineers, Standards Association of Piscataway, NJ
ITA	Independent Technical Authority
ITMRA	Information Technology Management Reform Act
IV&V	Independent Verification and Validation
MOTS	Modified-Off-The-Shelf
NESC	NASA Engineering and Safety Center
NPD	NASA Policy Directive
NPR	NASA Procedural Requirements
PDR	Preliminary Design Review
SCAMPI	Standard CMMI® Appraisal Method for Process Improvement
SEI	Software Engineering Institute
SW	Software
SWE	Software Engineering

APPENDIX D: Requirements Mapping Matrix

CLICK HERE FOR: MS Excel Version of Appendix D

				Class A	Class B	Class C	Class D	Class E	Class F	Class G	Class H
Training	*Software engineering training*	100	NASA Chief Engineer & Center Training Org	X	X	X	X	X	X	X	X
	Software training plan	101	Each Center	X	X	X	X	X	X	X	X
SW Document-ation Requirements	*SW Development/Mgt. Plan*	102	Project	X	X	P (Center)	P (Center)		X (not OTS)	P (Center)	
	SW Configuration Mgt. Plan	103	Project	X	P (Center)	P (Center)	P (Center)		X	P (Center)	
	SW Test Plan	104	Project	X	X	P (Center)	P (Center)		X	P (Center)	
	SW Maintenance Plan	105	Project	X	P (Center)				X	P (Center)	
	SW Assurance Plan	106	Project	X	X				X	P (Center)	
	Center SW Training Plan	107	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)
	Center SW Engineering Improve Plan	108	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)
	SW Requirements Spec.	109	Project	X	X	P (Center)	P (Center)		X	X	
	SW Data Dictionary	110	Project	X	P (Center)				X (not OTS)	X (not OTS)	
	SW Design Description	111	Project	X	X	P (Center)	P (Center)		X (not OTS)	X (not OTS)	
	Interface Design Description	112	Project	X	X	P (Center)			X (not OTS)	X (not OTS)	
	SW Change Request/ Problem Report	113	Project	X	X	P (Center)			X (not OTS)	X (not OTS)	
	SW Test Procedures	114	Project	X	X	P (Center)			X	X	
	SW Users Manual	115	Project	X	X				X (not OTS)	X (not OTS)	
	SW Version Description	116	Project	X	X	P (Center)	P (Center)		X (not OTS)	X (not OTS)	
	SW Metrics Report	117	Project	X	X	P (Center)			X (not OTS)	X (not OTS)	
	SW Test Report	118	Project	X	X	P (Center)			X	P (Center)	
	SW Inspection/Peer Review	119	Project	X	P (Center)				X (not OTS)	P (Center)	
Tailoring of Requirements	*Alternate requirement request*	120	Center	X	X	X	X	X	X	X	X
	Document approved alternate requirements	121	Center	X	X	X	X	X	X	X	X
Expertise of Warrant Authority	*Non-IT & Non Business*	122	NASA Chief Engineer & Chief Software Mission Assurance Officer	X	X	X	X	X			
	IT infrastructure & Business	123	CIO						X	X	X
Compliance	*Direction for Warrant Authority*	124	ITA Warrant Authority	X	X	X	X	X	X	X	X
	Compliance Matrix	125	Project	X	X	X	X	X	X	X	X
	Considerations for Waivers	126	ITA Warrant Authority	X	X	X	X	X	X	X	X
	Review of "P(Center)"	127	ITA Warrant Authority	X	X	X	X	X	X	X	X
	Compliance Records	128	ITA Warrant Authority	X	X	X	X	X	X	X	X
	Check compliance	129	NASA Chief Engineer	X	X	X	X	X	X	X	X

* See Requirement in NPR for full description

Light Grey shaded box indicates that the requirement cannot be waived by the project via the ITA (see Chapter 6)

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

X - Project 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 Center defined process

P (project) - As defined in the project or software development plan

Blank Space - Project is not required to meet the requirement

Note 1 - The scope of this requirement is contained in the source NPD or NPR

Note 2 - This requirement can only be waived by the OSMA ITA

Note 3 - For Class B software, in lieu of a CMM/CMMI certification by a developer, the project will conduct a software capability evaluation in the seven process area listed in SWE-032 and mitigate any risk, if deficient.

				Class A	Class B	Class C	Class D	Class E	Class F	Class G	Class H
SW Requirements Development	Document	49	Project	X	X	X	X	P (Center)	X	X	
	SW requirements	50	Project	X	X	X	X		X	X	
	Flow-down & derived req.	51	Project	X	X	X			X (not OTS)	P (Center)	
	Bi-directional trace	52	Project	X	X				X (not OTS)	P (Center)	
SW Requirements Management	Manage req. change	53	Project	X	X	X	X	P (Center)	X	X	
	Corrective action	54	Project	X	X				X	X	
	Requirements Validation	55	Project	X	X	X			X	X	
SW Design	Document design	56	Project	X	X	P (Center)			X (not OTS)	X (not OTS)	
	Architecture	57	Project	X	X	P (Center)	P (Center)		X (not OTS)	X (not OTS)	
	Detailed design	58	Project	X	X				X (not OTS)	X (not OTS)	
	Bi-directional trace	59	Project	X	X				X (not OTS)	X (not OTS)	
SW Implementation	Design --> code	60	Project	X	X	X	X		X (not OTS)	X (not OTS)	
	Coding standards	61	Project	X	X				X (not OTS)	X (not OTS)	
	Unit test	62	Project	X	X	X	P (Center)	P (Center)	X (not OTS)	X (not OTS)	
	Version Description	63	Project	X	X	P (Center)	X		X (not OTS)	X (not OTS)	
	Maintain Traceability	64	Project	X	X				X (not OTS)	X (not OTS)	
SW Testing	Plan, procedures, reports	65	Project	X	X	X	P (Center)		X	P (Center)	
	Perform testing	66	Project	X	X	X	X		X	P (Center)	
	Test for compliance	67	Project	X	X	X			X	P (Center)	
	Evaluate test results	68	Project	X	X	X	X		X	P (Center)	
	Doc. defect & track	69	Project	X	X	X	P (Center)		X	P (Center)	
	Models, simulations, tools	70	Project	X	X				X	P (Center)	
	Update plans & procedures	71	Project	X	X	X			X	P (Center)	
	Maintain Traceability	72	Project	X	X	X	X		X	P (Center)	
	Platform or Hi-Fidelity sim.	73	Project	X	X	X			X	P (Center)	
SW Operations, Maintenance, and Retirement	Document maint. plans	74	Project	X	X	X			X	P (Center)	
	Plan ops., maint. & retirement	75	Project	X	X	X			X	P (Center)	
	Implement plans	76	Project	X	X	X			X	P (Center)	
	Deliver software product	77	Project	X	X	X	X	P (Center)	X	X	
	As-built documentation	78	Project	X	X				X (not OTS)	X (not OTS)	
SW Configuration Management	Develop CM plan	79	Project	X	X	X	X		X	P (Center)	
	Track & evaluate changes	80	Project	X	X	X	P (Center)		X	P (Center)	
	Identify sw configuration items	81	Project	X	X	X	P (Center)		X	P (Center)	P (Center)
	Authorizing Changes	82	Project	X	X				X	P (Center)	P (Center)
	Maintain records	83	Project	X	X	X			X	P (Center)	P (Center)
	Configuration audits	84	Project	X	X				X	P (Center)	
	Implement procedures	85	Project	X	X	X	P (Center)	P (Center)	X	P (Center)	P (Center)
Risk Management	Continuous risk management	86	Project	X	X				X	P (Center)	
Peer Reviews	Requirements & Test Plans	87	Project	X	X	P (Center)			X (not OTS)	P (Center)	
	Checklist, criteria, & tracking	88	Project	X	X	P (Center)			X (not OTS)	P (Center)	
	Basic measurements	89	Project	X	X				X (not OTS)	P (Center)	
SW Measurement	Objectives	90	Project	X	X	X			X (not OTS)	P (Center)	
	SW measurement areas	91	Project	X	X	P (Center)			X (not OTS)	P (Center)	
	Collection & storage	92	Project	X	X	X			X (not OTS)	P (Center)	
	Analyze data	93	Project	X	X	P (Center)			X (not OTS)	P (Center)	
	Report analysis	94	Project	X	X	P (Center)			X (not OTS)	P (Center)	
	SW measurement system	95	Each Mission Directorate	X	X	X			X	X	
	Objectives & Procedures	96	Each Mission Directorate	X	X	X			X	X	
	Report results	97	Each Mission Directorate	X	X	X			X	X	
Best Practices	Agency process asset library	98	NASA Chief Engineer	X	X	X	X	X	X	X	X
	Identify applicable practices	99	Each Center	X	X	X	X	X	X	X	X

Section of NPR	Requirement Descriptor*	SWE Requirement #	Responsibility	Class A	Class B	Class C	Class D	Class E	Class F	Class G	Class H
Preface	Effective Date	1	General	X	X	X	X	X	X	X	X
Organizational Capability	Agency SW Initiative	2	NASA Chief Engineer	X	X	X	X	X	X	X	X
	Center Plan	3	Each Center	X	X	X	X	P (Center)	X	X	P (Center)
	Benchmark	4	NASA Chief Engineer	X	X	X	X	X	X	X	X
	SW Processes	5	Each Center	X	X	X	X	P (Center)	X	X	P (Center)
	List of Agency's programs & projects containing software	6	NASA Chief Engineer & Chief Software Mission Assurance Officer	X	X	X	X		X	X	
Compliance with Laws, Policies & Requirements	SW Disclosures	7	Project	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1
	Export Control	8	Project	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1
	External Release	9	Project	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1
	Security	10	Project	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1
	Disabilities	11	Project	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1
	Disabilities	12	Project	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1
SW Life Cycle Planning	SW Plan	13	Project	X	X	X	P (Center)	P (Center)	X	P (Center)	
	Execute Plan	14	Project	X	X	X	X	P (Center)	X	P (Center)	
	Cost Estimation	15	Project	X	X	X	P (Center)	P (Center)	X	P (Center)	
	Schedule	16	Project	X	X	X	P (Center)		X	P (Center)	
	Training	17	Project	X	X	X			X	P (Center)	
	Reviews	18	Project	X	X	X	X		X	P (Center)	
	Life Cycle	19	Project	X	X	X	P (Center)		X (not OTS)	P (Center)	
	SW Classification	20	Project	X	X	X	X	X	X	X	X
	SW Classification changes	21	Project	X	X	X	X	X	X	X	X
	SW Assurance	22	Project	X (Note 2)	X (Note 2)	P (project)			X	X	
	SW Safety	23	Project	X	X	X	X	X	X	X	X
	Plan Tracking	24	Project	X	X	X	P (Center)		X	P (Center)	
	Corrective Action	25	Project	X	X	X			X	P (Center)	
	Changes	26	Project	X	X	X			X	P (Center)	
Off The Shelf (OTS) SW	COTS, GOTS, MOTS	27	Project	X	X	X			X	P (Center)	
Verification & Validation	Verification planning	28	Project	X	X	X	P (Center)		X	P (Center)	
	Validation planning	29	Project	X	X	X	P (Center)		X	P (Center)	
	Verification results	30	Project	X	X	X	X		X	P (Center)	
	Validation results	31	Project	X	X	X	X		X	P (Center)	
Project Formulation	CMM L3 or CMMI L2	32	Project	X	X (Note 3)	P (Center)					
	Options for Acquisition	33	Project	X	X	X	X		X	X	
	Acceptance Criteria	34	Project	X	X	X	P (Center)		X	X	
	Supplier Selection	35	Project	X	X	X			X	X	P (Center)
	SW processes & tasks	36	Project	X	X	X	P (Center)		X (not OTS)	P (Center)	
	Milestone	37	Project	X	X	X	P (Center)		X (not OTS)	P (Center)	
	Acquisition planning	38	Project	X	X	X	P (Center)		X (not OTS)	P (Center)	
Government Insight	Insight into test	39	Project	X	X	P (Center)	P (Center)		X	P (Center)	
	Electronic access	40	Project	X	X	P (Center)			X	P (Center)	
	Open source	41	Project	X	X	P (Center)			X	P (Center)	
	Source code access	42	Project	X	X	P (Center)			X	P (Center)	
Supplier Monitoring	Track change request	43	Project	X	X	P (Center)			X	P (Center)	
	SW measurement data	44	Project	X	X	X	P (Center)		X	P (Center)	
	Joint audits	45	Project	X	X	X			X	P (Center)	
	SW schedule	46	Project	X	X	X	X		X	P (Center)	
	Traceability data	47	Project	X	X	P (Center)			X	P (Center)	
	Solicitation	48	Project	X	X	X	P (Center)		X	P (Center)	