TABLE OF CONTENTS

Preface ............................................................................................................................................ 2
P.1 Purpose .................................................................................................................................. 2
P.2 Applicability ............................................................................................................................ 2
P.3 Authority ................................................................................................................................. 3
P.4 Applicable Documents ........................................................................................................... 3
P.5 Measurement/Verification ...................................................................................................... 4
P.6 Cancellation ........................................................................................................................... 4
P.7 Training .................................................................................................................................. 4

Chapter 1. Center Level Requirements ...................................................................................... 5
1.1 LaRC Management and Organizational Units ....................................................................... 5
1.2 LaRC Software Engineering Process Group (SEPG) ............................................................ 5
1.3 LaRC Representative to the Software Working Group .......................................................... 6
1.4 Program and Project Managers ............................................................................................. 6

Chapter 2. Project-Level Requirements ....................................................................................... 7
2.1 General Requirements ........................................................................................................... 7
2.2 Software Classification and Safety Criticality ......................................................................... 7
2.3 Modifications to Requirements Mapping and Compliance Matrix ........................................... 8
2.4 NPR 7150.2 Compliance Matrix ............................................................................................. 9
2.5 Software Documents, Records, and Data ............................................................................... 9
2.6 Acquisition ............................................................................................................................ 11
2.7 Project Portfolios .................................................................................................................. 12

Chapter 3. Tailoring and Waivers ............................................................................................... 13
3.1 Compliance Designations and Tailoring .............................................................................. 13
3.2 Authority to Approve Tailoring ............................................................................................ 14

Appendix A. Supplemental Definitions ...................................................................................... 15
Appendix B. Abbreviations ........................................................................................................... 17
Appendix C. References ............................................................................................................... 18
Appendix D. LaRC Guidance on Compliance Matrices and Tailoring ........................................... 19
Appendix E. Recommended Document Contents ....................................................................... 29
Appendix F. LaRC Software Metrics Repository Data ............................................................... 45
Appendix G. Simplified NPR 7150.2B for Class D and E (Not Safety-Critical) ......................... 47
Preface

P.1 Purpose

a. This Langley Procedural Requirements (LPR) supplements NPR 7150.2 NASA Software Engineering Requirements for software classes A through E as defined in NPR 7150.2 Appendix D.

Note: The Office of the Chief Information Officer (OCIO) provides guidance on how Langley Research Center (LaRC) applies NPR 7150.2 to classes F through H.

b. This LPR clarifies, adapts, and implements select NPR 7150.2 requirements for Langley Research Center (LaRC) software activities. Those select NPR 7150.2 requirements are inherently governmental or apply to multiple software classes. This LPR flows the remaining requirements down to projects.

c. This LPR establishes and documents the LaRC software processes including augmentations and additions to NPR 7150.2 that support implementation of NPR 7150.2 at LaRC and integrate LaRC software activities with other LPRs and Langley Management System (LMS) Center Procedures (CPs). [SWE-005]

Note: The other LPRs and LMS-CPs cover technical authority, LMS waivers, software assurance, software release, and procurement.

P.2 Applicability

a. This LPR is applicable to all Class A through E software. The following activities and products are within scope:

(1) Software development, maintenance, operations, retirement, management, acquisition, assurance activities and services that are performed, created, or acquired by or for LaRC (hereafter referred to as projects); [SWE-001] [NPR 7150.2B:P.1]

(2) Computer programs, procedures, scripts, rules, and associated documentation and data pertaining to the development and operation of a computer system. This definition applies to software developed by NASA, software developed for NASA, commercial-off-the-shelf (COTS) software, Government-off-the-shelf (GOTS) software, modified-off-the-shelf (MOTS) software, reused software, auto-generated code, embedded software, the software executed on processors embedded in Programmable Logic Devices (see NASA-HDBK-4008), and open-source software components. [NPR 7150.2B:A]

Note: This LPR divides software into “developed software” and “non-developed software” as defined in Appendix A. Developed software encompasses new software and software modifications developed by or for NASA. Non-developed software covers any existing software that the project obtains such as COTS, GOTS, MOTS, reused, heritage, and open source software.

b. This LPR applies to the personnel, programs, projects, and tasks at LaRC, including contractors, grant recipients, or parties to agreements to the extent specified in their respective contracts or agreements. [NPR 7150.2B:P.2.a]

c. This LPR is made applicable to contractors, grantees, or external partners through contractual clauses, specifications, or statements of work in conformance with the NASA Federal Acquisition Regulation (FAR) Supplement or through grants or agreements. [NPR 7150.2B:P.2.a Note]

d. Exclusions:

(1) The gate level structure (hardware aspects) of field programmable gate arrays (FPGA), complex programmable logic device (CPLD), system-on-chip (SoC), and application-specific integrated circuits (ASIC) are not subject to this LPR. Processor cores can be incorporated into complex electronic devices (FPGA, ASIC, SoC). Though the processor core falls under this exclusion, machine or byte code that runs on the processor core is software, and the LPR applies to the code. This remains true in the case where the code is stored on-chip.

Note: Due to differing engineering processes, the above devices will be subject to a tailored flight hardware development plan using guidance from NASA-HDBK 8739.23, NASA Complex Electronics Handbook for Assurance Professionals, until further Agency or Center direction is provided.
(2) Information technology systems as defined in Classes F, G, and H. These projects follow OCIO guidance for NPR 7150.2 conformance. [NPR 7150.2B:Appendix D]

(3) Stand-alone desktop applications (e.g., word processing programs, project scheduling software, presentation programs). [NPR 7150.2B:3.9.1]

(4) This LPR does not apply if solely acquiring non-developed software for unmodified use outside the context of a NASA system (i.e., acquiring a standalone, completed software product that will not be modified and will not be included within a NASA system; and whose output is not used to engineer, verify, validate, operate, or maintain a NASA system). [NPR 7150.2B:3.9.1]

Note: Contact the LaRC Office of Procurement to procure COTS products.

(5) This LPR does not apply to non-safety critical, standalone software that:
(a) Has no anticipated delivery, and;
(b) Is not the subject of a publication or delivered/published analyses, and
(c) Is not included in a system that is being delivered, and
(d) Will not be used to make decisions on Class A, B, C, or D systems.

For questions about eligibility of this exclusion, contact your Software Engineering Process Group (SEPG) Representative (see paragraph 1.1.1); the list of SEPG Representatives is available at http://sw-eng.larc.nasa.gov/.

(6) This LPR does not retroactively apply to software engineering activities started before May 14, 2013.

e. In this LPR, all mandatory actions (i.e., requirements) are denoted by statements containing the term "shall." The terms "may" or "can" denote discretionary privilege or permission, "should" denotes a good practice, "will" denotes expected outcome, and "are" or "is" denotes descriptive material. Additionally, any text in grey is informative. [NPR 7150.2B:P.2.f]

f. References of the form '[SWE-ddd]' (where 'ddd' is a three-digit, zero-padded whole number) refer to software engineering (SWE) requirements numbers in NPR 7150.2; they are included in this LPR to show traceability and compliance to NPR 7150.2 requirements. [SWE-140]

g. Requirements established by this LPR are followed with a requirements number of the form '[LSWE-ddd]' where 'ddd' is a three-digit, zero-padded whole number.
Note: Most LSWE requirements are specializations of the referenced SWE requirements in a LaRC context and are not additional requirements above those in NPR 7150.2.
Note: The following LSWE numbers are reserved: LSWE-042.

h. All other references between brackets identify the source of the preceding text.
i. All document citations are intended to reference the latest revision of the document unless otherwise noted. [NPR 7150.2B:P.2.h]
j. In this LPR, "procedural requirements" are the body of requirements from the NASA Online Directives Information System (NODIS) and the Langley Management System (LMS) that apply to software activities. "Procedural requirements" also includes requirements from NASA standards when those standards are invoked from applicable documents in NODIS or LMS. See P.4 Applicable Documents for the list of applicable documents.

P.3 Authority

a. NPD 1280.1, NASA Integrated Management System Policy.

Note: NPDs and NPRs are found in the NASA Online Directives Information System (NODIS) at: http://nodis3.gsfc.nasa.gov/.

P.4 Applicable Documents

The latest versions of the following documents are to be applicable. In the event of conflict among the top-level directives and one or more lower-level directives or procedures, the information provided in the top-level directive takes precedence. In the event of a conflict among documents at the same level, consult the designated Technical Authority for resolution (see paragraph 2.1.3).

Note: NPDs and NPRs are found in the NASA Online Directives Information System (NODIS) at: http://nodis3.gsfc.nasa.gov/. LAPDs, LPRs, and LMS-CPs are found on the Langley Management System web site: https://lms.larc.nasa.gov.


Page 3 of 53

Verify the correct revision before use by checking the LMS Web site.
b. LAPD 1150.2, Councils, Boards, Panels, Committees, Teams, and Groups.
c. LAPD 2810.1, Security of Information Technology.
e. LPR 7120.4, Langley Research Center Technical Authority Implementation Plan.
f. LMS-CP-1724, Software Release.
g. LMS-CP-7151, Obtaining Waivers for Langley Management System (LMS) Requirements.
h. LMS-CP-4754, Software Assurance (SA) for Development and Acquisition.

P.5 Measurement/Verification

a. The NASA Headquarters (HQ) Office of the Chief Engineer (OCE) authorizes appraisals against this LPR to check compliance. [SWE-129]
b. Compliance with this LPR is documented in the software Compliance Matrix maintained by each software project (see section 2.5 details).

P.6 Cancellation

a. LPR 7150.2A, LaRC Software Engineering Requirements, May 14, 2013..
e. LMS-CP-7150.6, Class E Software, Revision A, May 14, 2013.

P.7 Training

a. For answers to questions or to request training on this LPR, send an email to: larc-dl-support-sepg-help or call the LMS Software procedure help desk phone number provided at https://sw-eng.larc.nasa.gov/.
b. Individuals in Core Resource Unit Directorates (CRUDs) may also contact their Software Engineering Process Group (SEPG) Representative for assistance. The SEPG representatives are listed at https://sw-eng.larc.nasa.gov/.

/s/ Clayton Turner February 8, 2018
Center Deputy Director Date

DISTRIBUTION
Approved for public release via the Langley Management System; distribution is unlimited.
Chapter 1. Center Level Requirements

1.1 LaRC Management and Organizational Units

1.1.1 Each Engineering Director shall appoint a representative to the Software Engineering Process Group (SEPG). [LSWE-001] [SWE-003]
Note: The SEPG is a LaRC group responsible for defining and implementing LaRC’s plan to advance in-house engineering capability, including maintenance of this LPR. The SEPG Representative is a recognized software engineering expert within the directorate. In addition to their SEPG duties, the SEPG Representatives are also responsible for assisting their directorates in interpretation and application of the procedural requirements and are a resource available to the NASA Software Leads and the designated Engineering Technical Authority(ies) to advise on software classification and tailoring requests.

1.1.2 The Chief Engineer shall appoint the LaRC representative and an alternate to the Agency Software Working Group (SWG). [LSWE-002] [SWE-003]
Note: The SWG is a Headquarters group that assists the NASA Chief Engineer in defining, maintaining, and executing the NASA Software Engineering Initiative to advance software engineering practices within the Agency. The LaRC SWG Representative also acts as the SWG point of contact on the SEPG and manages funds from Headquarters to support software engineering initiatives at LaRC.

1.1.3 The Head of the Mission Assurance Branch shall designate the Center Software Assurance Manager. [LSWE-003] [SWE-003]
Note: The Center Software Assurance Manager performs software assurance for select procedural requirements as defined in this LPR, represents the software assurance discipline on the Software Engineering Process Group (SEPG), and acts as a point of contact between the project and the designated Safety and Mission Assurance (SMA) Technical Authority.

1.1.4 The LaRC Training, Development, and Employee Relations Branch shall provide and fund training to advance software engineering practices and software acquisition. [LSWE-004] [SWE-100]

1.2 LaRC Software Engineering Process Group (SEPG)

Refer to LAPD 1150.2 Councils, Boards, Panels, Committees, Teams, and Groups for the SEPG charter.

1.2.1 The SEPG shall maintain this LPR. [LSWE-005] [SWE-005] [SWE-003]

1.2.2 The SEPG shall maintain, staff, and implement the Center Plan for LaRC Software Process Improvement to continually advance its in-house software engineering capability and monitor the software engineering capability of NASA’s contractors. [LSWE-006] [SWE-003]
Note: Go to http://sw-eng.larc.nasa.gov/ to see the latest plan.

1.2.3 The SEPG shall establish and maintain the LaRC Software Metrics Repository for entry and storage of software project and cost measurements from the Center’s Class A, Class B, Class C, and safety-critical software projects. [LSWE-007] [SWE-091] [SWE-092] [SWE-142]
Note: More information on the LaRC Software Metrics Repository is found in Appendix F and at https://sw-eng.larc.nasa.gov/metrics-collection/.

1.2.4 The SEPG shall establish and maintain the LaRC software measurement program, which utilizes the LaRC Software Metrics Repository to monitor software engineering capability, improve software quality, and track software engineering improvement. [LSWE-008] [SWE-092]
Note: This program is documented in the LaRC Software Measurement Description hosted at https://sw-eng.larc.nasa.gov/supporting-products/.

1.2.5 On an annual basis, the SEPG shall analyze data from the Langley Software Metrics Repository and report results to the LaRC Chief Engineers Board. [LSWE-009] [SWE-094]

1.2.6 The SEPG shall maintain and implement a software training plan to advance LaRC’s in-house software engineering capability and as a reference for its contractors. [LSWE-010] [SWE-101]

1.2.7 As requested by HQ OCE, the SEPG shall solicit software engineering process assets from the organizational units for inclusion in the Agency-wide process asset library. [LSWE-052] [SWE-144]
1.3 LaRC Representative to the Software Working Group

1.3.1 The LaRC SWG Representative shall report on the status of the Center’s software engineering discipline, as applied to LaRC projects, to the NASA Office of the Chief Engineer and relevant technical authorities as requested. [LSWE-011] [SWE-099]
Note: This report is typically presented annually at a face-to-face meeting of the SWG.

1.3.2 The LaRC SWG Representative shall maintain a reliable list of LaRC programs and projects containing Class A, B, C, and D software. [LSWE-012] [SWE-006]
Note: Each project is required to supply information on software tasks to the LaRC SWG Representative in paragraph 2.5.4.

1.4 Program and Project Managers

The Program and Project Managers reside in the Product Unit Directorates (PUDs) as defined in the NASA Langley Strategic Framework (https://strategy.larc.nasa.gov/).

1.4.1 The Program and Project Managers shall ensure that plans, resources, procurements, and agreements will support compliance with requirements marked with an ‘X’ in NPR 7150.2 Appendix C. [LSWE-013] [SWE-140]

1.4.2 The software management process requires the understanding and application of laws and NASA policy requirements that impact the development, release, and/or maintenance of software. NPR 7150.2 section 2.1.5 lists requirements outside of NPR 7150.2 that may affect software projects. These requirements cover invention disclosure, intellectual property rights, software release, technology transfer, information security, accessibility for individuals with disabilities, Independent Verification & Validation (IV&V), human rating requirements, and problem data exchange between government and industry on spaceflight projects. Program and Project Managers will ensure these requirements are implemented as applicable.
Chapter 2. Project-Level Requirements

2.1 General Requirements

2.1.1 Projects shall fully comply with the requirements in this LPR and the requirements marked ‘X’ in NPR 7150.2 Appendix C. [LSWE-014] [SWE-139]
Note: Requirements may be tailored down or tailored out with technical authority approval per Chapter 3.
Note: For the convenience of Class D projects and Class E projects that are not safety-critical, the NPR 7150.2B requirements for those classifications have been extracted and presented as a notional process flow in Appendix G. These projects can use Appendix G as a substitute for NPR 7150.2B.

2.1.2 The line manager of the LaRC organization responsible for the software task shall assign the role of NASA Software Lead. [LSWE-015]
Note: This LPR uses the role of NASA Software Lead to encompass the individual or individuals responsible for project-level requirements that this LPR assigns to the Government. The NASA Software Lead may rely on assistance from contractors to accomplish assigned requirements but is the responsible official for content and approval of results.

2.1.3 The Engineering Technical Authority for the software task is designated per LPR 7120.4. [SWE-122]
Note: For most software tasks, the engineering technical authority flows from the Center Director to the Engineering Directors (i.e. the directors of the Center Operations Directorate, Research Services Directorate, Research Directorate, Science Directorate, Systems Analysis and Concepts Directorate, and Engineering Directorate). An Engineering Director can further designate an engineering technical authority for a program or project or for an engineering discipline (e.g. a discipline lead engineer). LPR 7120.4 designates Branch Heads as the lead discipline engineers for the branch and permits the Engineering Director to delegate the engineering technical authority to the Branch Head. [LPR 7120.4: 7.3 & 7.7]

2.1.4 The Safety and Mission Assurance (SMA) Technical Authority for the software task is designated per LPR 7120.4.
Note: Involvement of the SMA Technical Authority is rarely needed when the software is Class D or Class E and not safety-critical.

2.2 Software Classification and Safety Criticality

2.2.1 The NASA Software Lead shall classify each system and subsystem containing software in accordance with the software classification definitions for Classes A, B, C, D, and E specified in NPR 7150.2 Appendix D. [LSWE-016][SWE-020]
Note:
- The NASA Software Lead classifies the system or subsystem in which the software operates, not the software itself. (See definition of system and subsystem in NPR 7150.2 Appendix A.) Moreover, a given system or subsystem is assigned a single class.
- Classifications are defined by intended use. Software is only a logical description of a function and requires the remainder of the system (e.g., hardware, other software, and data) to realize that function in the context of an intended use. Thus, the same software item can be classified differently in different systems. [NPR 7150.2B: Appendix C]
- When non-developed software is included within a NASA system or subsystem, the non-developed software is assessed and classified as part of that system or subsystem. [NPR 7150.2B: 3.9.2]
- Some projects may contain multiple systems and subsystems having different software classes. [NPR 7150.2B: P.2.1] In this circumstance, a project can conjointly develop the software using the processes and associated work products established for the highest class. The project does not have to segregate software items by class and use different processes, work products, or teams for each class.
- The number of applicable procedural requirements and their associated rigor are scaled back for lower software classes and software designated as not safety-critical. [NPR 7150.2B: Appendix C]
2.2.2 If more than one software class matches a given system or subsystem, then the NASA 
Software Lead shall assign the higher of the classes to the system or subsystem. [LSWE-017] 
[NPR 7150.2B:Appendix C] 
Note: Classes are in descending order from A to E.

2.2.3 The NASA Software Lead shall determine safety-criticality of the software in accordance with 

2.2.4 Following LMS-CP-4754 Software Assurance (SA) for Development and Acquisition, the NASA 
Software Lead shall notify the Center Software Assurance Manager of the software activity, the 
associated software classification(s), and the safety-critical determination. [LSWE-019] [SWE-022] 
[NPR 7150.2B:Appendix C] 
Note: The Center Software Assurance Manager will perform an independent assessment of 
software classification and safety-critical determination and document it in a Software 
Assurance Classification Report (SACR). The NASA Software Lead and the Center Software 
Assurance Manager will reach agreement on classification and safety-critical determination or 
will elevate disagreements using the dissenting opinion section of LPR 7120.4, Langley 
Research Center Technical Authority Implementation Plan.

2.2.5 If a system or subsystem evolves to a higher software classification, or changes from non-
safety-critical to safety-critical, then the NASA Software Lead shall return to this LPR 7150.2 to 
repeat completion of sections 2.2 through 2.5. [LSWE-020] [SWE-006] [SWE-021] [SWE-125] 
Note: As a result the NASA Software Lead will: 
- Complete or amend the NPR 7150.2 Compliance Matrix for the new software class or 
safety-critical determination. 
- Obtain approval for tailoring (including previously approved tailoring) from the technical 
authority for the new software class or safety-critical determination. 
- Ensure the project updates plans to fulfill the added requirements specified for the higher 
software Class and/or safety criticality per NPR 7150.2. 
- Provide new or updated software inventory data for the project

2.3 Modifications to Requirements Mapping and Compliance Matrix 
NPR 7150.2B Appendix C maps the NPR's requirements with project-level responsibility to each 
software class. This section documents LaRC-specific augmentations to the mapping. 

2.3.1 LaRC projects shall comply with the augmentations to NPR 7150.2 Appendix C presented in 
Table 2.1: [LSWE-021]

<table>
<thead>
<tr>
<th>Software Class</th>
<th>Requirement</th>
<th>Old Mapping</th>
<th>New Mapping</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class A, Class B, Class C, or Safety-Critical</td>
<td>LSWE-031</td>
<td>None</td>
<td>X</td>
<td>The LaRC Software Metrics Repository is the Center's implementation of SWE-091 and SWE-142.</td>
</tr>
<tr>
<td></td>
<td>LSWE-030</td>
<td>None</td>
<td>X</td>
<td>This is the Center's implementation of SWE-006.</td>
</tr>
<tr>
<td>Class D</td>
<td>LSWE-030</td>
<td>None</td>
<td>X</td>
<td>This is the Center's implementation of SWE-006.</td>
</tr>
<tr>
<td></td>
<td>SWE-035</td>
<td>*(S/C only)</td>
<td>X</td>
<td>Projects are required to follow LMS procedures for procurement regardless of class or safety-criticality. See LMS-CP-4501.</td>
</tr>
<tr>
<td>Class E</td>
<td>SWE-033</td>
<td>&lt;blank&gt;</td>
<td>X</td>
<td>When a project acquires software, the project is required to contact the Office of Procurement regardless of class or safety-criticality.</td>
</tr>
<tr>
<td></td>
<td>SWE-035</td>
<td>&lt;blank&gt;</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Page 8 of 53

Verify the correct revision before use by checking the LMS Web site.
### 2.4 NPR 7150.2 Compliance Matrix

#### 2.4.1 The NASA Software Lead shall produce a NPR 7150.2 Compliance Matrix, in accordance with Appendix D, for each class of software within the project. [LSWE-022] [SWE-125]

Note: When contracts, grants, or agreements will be used to accomplish the software task, the compliance matrix documents the division of responsibility for the procedural requirements between LaRC and third parties. The NASA Software Lead should make an initial cut at this division during planning of the acquisition or agreement. This division should be finalized with the signing of the contract, grant, or agreement.

Note: Because the compliance matrix documents the division of responsibility for procedural requirements between LaRC and third parties and because civil-service Technical Authorities must approve tailoring in the compliance matrix, this LPR explicitly makes the compliance matrix an inherently governmental activity.

#### 2.4.2 The NASA Software Lead shall use the NPR 7150.2 Compliance Matrix to document any tailoring of procedural requirements (i.e., NPR 7150.2 or LMS requirements); content requirements for tailoring requests are documented in Appendix D. [LSWE-023] [SWE-121]

#### 2.4.3 The NASA Software Lead shall obtain approvals from the following officials for the NPR 7150.2 Compliance Matrix:

a. NASA Software Lead.
b. NASA Software Lead’s line manager.
c. Designated Engineering Technical Authority per Chapter 3 when tailoring. [SWE-145]
d. Designated SMA Technical Authority when applicable per section 3.2. [SWE-145]
e. Center Health and Medical Technical Authority when applicable per LPR 7120.4. [SWE-145]

Note: This is uncommon. Most software developed by LaRC does not have a use that falls under the authority of the Health and Medical Technical Authority.

Note: The NASA Software Lead should seek approval for tailoring prior to the finalization or approval of solicitations, agreements, and/or plans that include or implement the tailoring.

#### 2.4.4 When tailoring is approved, the designated Engineering Technical Authority will retain a copy of the approved NPR 7150.2 Compliance Matrix as an archived, retrievable record. [SWE-126]

Note: Headquarters or the Center may request access to approved NPR 7150.2 Compliance Matrices when conducting audits or reviews or to inform future process improvements.

#### 2.4.5 The NASA Software Lead shall provide a copy of the approved NPR 7150.2 Compliance Matrix to the Center Software Assurance Manager for review and concurrence. [LSWE-025]

Note: The Center Software Assurance Manager assures that the compliance matrix complies with this LPR and NPR 7150.2.

#### 2.4.6 To assure compliance with this LPR and as specified in LMS-CP-4754 Software Assurance (SA) for Development and Acquisition, the Mission Assurance Branch shall perform spot check audits between the project and its NPR 7150.2 Compliance Matrix when the software is not safety-critical and is Class D. [LSWE-026] [SWE-022]

#### 2.4.7 For all software that is Class A, Class B, Class C, or safety-critical, the Mission Assurance Branch shall assure compliance with requirements and standards as specified in LMS-CP-4754, Software Assurance (SA) for Development and Acquisition. [LSWE-027] [SWE-022]

### 2.5 Software Documents, Records, and Data

#### 2.5.1 The designated Engineering Technical Authority shall define the content requirements for software documents or records to be produced by the project, with the exception of software assurance documents and records. [LSWE-028] [SWE-153]

Note: Appendix E provides Center recommendations for content requirements mapped to

---

Verify the correct revision before use by checking the LMS Web site.
software class. The Engineering Technical Authority can simply approve the use of Appendix E for a software task, augment Appendix E, or create an alternative set of content requirements.

Note: A LaRC organization can define a standard set of documents and content for a project portfolio (see section 2.7) and obtain one-time approval for those standards from the Engineering Technical Authority.

2.5.2 For safety-critical software, the designated Engineering Technical Authority shall coordinate the content requirements for software documents or records with the designated Safety and Mission Assurance (SMA) Technical Authority. [LSWE-029]

Note: The Center SMA Technical Authority assesses the adequacy of software documents and records to support safety engineering, assessment, analysis, and certification.

2.5.3 The designated SMA Technical Authority will define the content requirements for software documents and records within the software assurance discipline.

2.5.4 For software classifications Class A through Class D, the NASA Software Lead shall provide the following project data to the NASA Software Working Group Representative on project startup: [LSWE-030] [SWE-006]

a. Name of project
b. Work breakdown structure (WBS) number
   Note: This is the number, within the Agency's financial system, of the WBS that funds the software activity. For example, this is the labor code that appears in WebTADS.
c. Name and contact information for the NASA Software Lead
d. Name of the development organization(s)
e. Name of the software assurance organization(s), if applicable
f. Primary lifecycle methodology
g. Current lifecycle phase
h. For each software item, provide the following information:
   1) Name of item
   2) Estimate of software size as KSLOC (thousand source lines of code)
   3) Software class and safety-critical determination
   4) The primary programming language

Note: The intent is not to identify and track each piece of software but to track and characterize funded software efforts. Therefore, the Center encourages rolling together software tasks with shared funding or to report project portfolios rather than individual software tasks.

Note: The NASA Software Working Group Representative may ask the NASA Software Lead for updated data in response to data calls from Center management or from Headquarters.

2.5.5 When the software class is Class A, Class B, or Class C, or the software is safety-critical, the NASA Software Lead shall enter the project data for the LaRC Software Metrics Repository as described in Appendix F. [LSWE-031] [SWE-091] [SWE-142]

Note: Data entry for the LaRC Software Metrics Repository occurs at multiple points in the project's lifecycle from startup to retirement. It is not a one-time action.

Note: The NASA Software Lead may need the software team to deliver the project data. Therefore, the NASA Software Lead should ensure that data delivery is included in software plans and agreements.

2.5.6 The project will submit a New Technology Report (NTR) for developed software. NTRs are submitted online via http://invention.nasa.gov. [NPR 2091.1B][NASA FAR Supplement: 1852.227]

NOTE: The project should begin the process of submitting an NTR when the new software or software feature is first conceived and should maintain the NTR throughout the software lifecycle. The NTR asks for lifecycle-specific information, which may require significant effort for software having a lengthy development history.

NOTE: When software has previously been reported using an NTR, a new NTR is necessary for future revisions of the software except when the revision contains only bug fixes or minor enhancements per NPR 2210.1. [NPR 2210.1C 1.8.4, 1.8.5, and 2.2.2.1]

NOTE: The NTR is a pre-requisite for releasing software per LMS-CP-1724. For software previously released per LMS-CP-7124, the Center Software Release Authority has the flexibility to accept a minor enhancement for release without receiving a new NTR. [NPR 2210.1C 1.8.4, 1.8.5, and 2.2.2.1]
2.5.7 For software classifications Class A through Class D, the NASA Software Lead shall evaluate software for potential reuse by other projects across the Agency and contribute reuse candidates to the Agency Software Catalog. [SWE-148]

NOTE: To add software to the Agency Software Catalog, submit the software through the software release process per LMS-CP-1724.

NOTE: If one purpose of the project is to develop reusable software, then the NASA Software Lead must consider the needed intellectual property rights to the software when making decisions during formulation including but not limited to make versus buy decisions and working with the Office of the Chief Counsel to include appropriate intellectual property rights language in any agreements used to develop some or all of the software.

2.5.8 For software classifications Class A through Class D, the NASA Software Lead shall define and document the acceptance criteria and conditions for the software. [SWE-034]

Guidance:

a. Examples of conditions are operating under maximum loads such as number of simultaneous users or peak bandwidth capacity, and operational modes such as startup, operations, shutdown, and maintenance.

b. For software developed using suppliers, the acceptance criteria and conditions will be defined and documented in the solicitation and final agreement. See section 2.6.10.

c. For software developed by civil servants, acceptance criteria and conditions can be documented as part of the requirements documentation or as part of software plans.

2.6 Acquisition

This section covers acquisition of software products or services performed by LaRC. When LaRC permits a supplier to perform software acquisition on behalf of the Government, then the NASA Software Lead will flow down the procedural requirements covering software acquisition into the supplier agreement as part of paragraph 2.6.5.

2.6.1 The NASA Software Lead shall assess options for software acquisition versus development (make versus buy assessment). [LSWE-032] [SWE-033]

Note: Assess the risk, cost, and benefits of each of the options listed below: [NPR 7150.2A:3.12.2]

1. Acquire a non-developed (e.g., COTS) software item that satisfies the requirement.
2. Develop the software item or obtain the software service internally.
3. Develop the software item or obtain the software service through a contract or other agreement.
4. Enhance an existing software item or service.
5. Reuse an existing software item or service.

2.6.2 To procure software items or services, the NASA Software Lead will contact the LaRC Office of Procurement. [SWE-035]

2.6.3 When non-developed software (including open source software) will be included in a NASA system or subsystem, the NASA Software Lead shall consult with the Office of Chief Counsel to assess whether associated intellectual property rights and licenses are agreeable to LaRC and appropriate to meet the needs of the project. [LSWE-033] [NPR 2210.1C:1.9.3]

2.6.4 When LaRC uses procurement to develop or acquire a software item that LaRC intends to release (per LMS-CP-1724), the NASA Software Lead shall consult with the Office of Chief Counsel and the Office of Procurement to assure that LaRC receives the necessary intellectual property rights and licenses to the acquired software item. [LSWE-034]

Note: This activity can apply to software items that will be released to another Center. For example, the standard license for a non-developed software item that is included in the NASA system may not grant the rights to transfer the license to the other Center.

2.6.5 The NASA Software Lead shall determine which procedural requirements that LaRC will flow down to the supplier and shall identify those requirements in the NPR 7150.2 Compliance Matrix, the solicitation, and the final agreement. [LSWE-035] [SWE-036][SWE-038]

2.6.6 When the software is Class A, Class B, Class C, or safety-critical, the NASA Software Lead and the Center Software Assurance Manager shall determine the NASA-STD-8739.8 requirements to be performed by the supplier. [LSWE-036]
Note: NASA-STD-8739.8 Appendix C contains a Requirements Compliance Matrix that may be used to document the division of responsibility between LaRC and the supplier.

2.6.7 When the software is safety-critical, the NASA Software Lead and the designated SMA technical authority shall determine the NASA-STD-8719.13 requirements to be performed by the supplier. [LSWE-037]
Note: NASA-STD-8719.13, Appendix B-2 contains a Requirements Compliance Matrix that may be used to document the division of responsibility between LaRC and the supplier.

2.6.8 The NASA Software Lead shall define the technical work that the supplier will perform and shall document the technical work in the solicitation and final agreement. [LSWE-038] [SWE-050]
Note: On some projects, the scope of the technical work is defined in the agreement, but the development of detailed software requirements is a technical activity performed under the agreement that is the joint responsibility of LaRC and the supplier.

2.6.9 The NASA Software Lead shall, in consultation with the Engineering Technical Authority, determine and define the software plans and work products to be delivered by the supplier. [LSWE-039] [SWE-013] [SWE-153]
Note: Appendix E provides recommended software documents and content mapped to the software classes. The appendix can be used directly with approval of the Engineering Technical Authority.

2.6.10 For software Class A through Class D, the NASA Software Lead shall document the acceptance criteria and conditions for deliverable software items and services in the solicitation and the final agreement. [LSWE-040] [SWE-034]

2.6.11 The NASA Software Lead shall assure that the solicitation and final agreement address the following NPR 7150.2 requirements, as applicable for the software class and safety-critical determination: SWE-037, SWE-039, SWE-040, SWE-041, SWE-042, SWE-043, SWE-046, SWE-047, and SWE-077. [LSWE-041]
Note: These requirements cover a) reviews, audits, or other joint activities to be performed under the agreement, b) LaRC access to software engineering data and work products while the agreement is active, and c) deliverable software items, data, and work products to support LaRC insight and oversight activities, operations, and maintenance.

2.6.12 The NASA Software Lead shall assure that the solicitation and final agreement include delivery of measurements required for the software inventory (see paragraph 2.5.4) and the LaRC Software Metrics Repository (see paragraph 2.5.5). [LSWE-042]

2.7 Project Portfolios

2.7.1 When a line organization manages multiple software tasks within the same domain, the line organization can combine those tasks into a project portfolio.

2.7.2 The line organization shall manage the project portfolio using either a) an umbrella software management plan or b) a documented software policy and organizational standard software engineering process. [LSWE-043]
Note: Under option (a), the software tasks are managed concurrently by the line organization using a unified process defined in an organization-level software management plan. Option (b) allows a line organization to manage each task within the portfolio as a distinct project with its own software management plan; however, each project follows an organizational standard for software engineering that is pre-tailored and pre-approved for the domain of the project portfolio. The software policy defines the domain for the project portfolio and identifies the organizational standards applicable to the portfolio; often the software policy also describes the organization’s roles and responsibilities for software engineering of the portfolio.

2.7.3 The software management plan or software policy shall define the domain (i.e., scope) of the project portfolio. [LSWE-044]

2.7.4 Software tasks within a project portfolio may share a NPR 7150.2 Compliance Matrix.
Note: The designated Technical Authorities approve tailoring that is documented in the compliance matrix for the project portfolio and are not required to revalidate the compliance matrix for each software task added to the project portfolio. A software task within a project portfolio is still free to add further tailoring by creating an individual NPR 7150.2 Compliance Matrix and obtaining approvals per Chapter 3.
2.7.5 For each software task added to the project portfolio, the NASA Software Lead shall perform classification per section 2.2 and shall notify the Center Software Assurance Manager of intent to add the software task to the project portfolio. [LSWE-045]

Note 1: Whether a project portfolio covers one class or multiple classes, NPR 7150.2 requires that each NASA system or subsystem containing software is classified and that software assurance independently assesses classification.

Note 2: The Center Software Assurance Manager assures that NPR 7150.2 Compliance Matrices comply with this LPR and NPR 7150.2 (see section 2.4.4). Notifying the Center Software Assurance Manager of additions to the project portfolio informs the Center Software Assurance Manager that the software task falls under the compliance matrix of the project portfolio.

2.7.6 Within this LPR excluding this section, "project portfolio" may substitute where the term "project" is used unless otherwise excluded in the text.

Chapter 3. Tailoring and Waivers

Tailoring covers requests to partially implement a procedural requirement, omit a procedural requirement, or replace a procedural requirement with an alternative practice. Tailoring, as described in this LPR, covers only those procedural requirements within the project's scope of responsibility. Procedural requirements not in the scope of a project's responsibilities per NPR 7150.2 Appendix C and section 2.3 of this LPR, require an LMS waiver per LMS-CP-7151 Obtaining Waivers for Langley Management System (LMS) Requirements.

3.1 Compliance Designations and Tailoring

3.1.1 The NASA Software Lead shall use one of the following compliance designators to designate the level of compliance that the project will employ for procedural requirement: [LSWE-046]

a. FC – full compliance
b. T – tailored.
c. NA – not applicable.

3.1.2 The full-compliance designator ‘FC’ indicates that the project intends to meet the intent of the procedural requirement.

Note: 'Meet the intent' is interpreted in the context of the software classification and safety-critical determination. The project is expected to address the complete text of the statement, but lower classes may use informal methods to satisfy the requirement. Appendix D also provides specialized and reduced-scope versions of select SWE requirements for some lower software classes to refine “meet the intent” when the full text of the SWE cannot be applied to the class.

Note: This LPR provides a fully compliant implementation for some SWE requirements allocated to the projects. These requirements are listed in Appendix D and can be marked FC in the project's compliance matrix when the project fully complies with this LPR.

3.1.3 The tailoring designator "T" indicates that the project intends to partly fulfill ('tailor down'), disregard ('tailor out'), or replace the procedural requirement.

Note: Tailoring requires technical authority approval per this chapter. Appendix D requires that tailoring requests include an engineering justification to assist the technical authority in assessing how the tailoring changes technical risk on the project. Tailoring can negatively, neutrally, or positively impact technical risk. Technical authorities will give more scrutiny to tailoring that increases risk and may ask the project to add mitigations against the increased risk. Tailoring out a procedural requirement is often done either because the cost of the requirement outweighs its benefit on the project or because unique characteristics of the project prevent application of the requirement. For the latter, the requirement is often described as 'not applicable;' however, the 'not applicable' designator of the Compliance Matrix is reserved for a more restrictive use as described in the next paragraph. The more general uses of 'not applicable' should be marked as 'T' (tailored) because they require review and confirmation by the technical authority per SWE-126.

3.1.4 The not-applicable designator 'NA' is used when the project does not meet the restricted applicability of a procedural requirement as documented in Appendix D.2.

Verify the correct revision before use by checking the LMS Web site.
Note: Some procedural requirements have restricted applicability as documented in NPR 7150.2. When a project does not meet the restriction, the requirement may be marked 'NA' and disregarded without obtaining approval of the technical authority. The NA designation is not a generic 'not applicable' designation and is not to be used when a procedural requirement is not applicable due to unique circumstances of a project. This situation is to be treated as a 'tailor out' of the requirement under the 'T' designation. SWE-126 requires such 'tailor out' determinations to be reviewed and confirmed by the technical authority.

3.2 Authority to Approve Tailoring

3.2.1 The designated Engineering Technical Authority for the software task (see paragraph 2.1.3) has authority to approve tailoring of procedural requirements mapped to a software class (see Appendix D) with the following exceptions: [SWE-126][SWE-150][NPR 7150.2B: Appendix C]

a. SWE-020, SWE-021, SWE-132, SWE-133, SWE-125, SWE-139, and SWE-145
   Note: These requirements have been embedded into the procedures of this LPR and its supporting LMS Center Procedures. Consequently, this LPR does not delegate tailoring approval for these requirements from the Center Director to lower level technical authorities. To tailor these requirements requires an LMS waiver per LMS-CP-7151.

b. SWE-032 and SWE-141 [NPR 7150.2B Appendix C]
   Note: NPR 7150.2 does not delegate tailoring approval for these requirements to the Center Director. To tailor these requirements, the project first submits an LMS waiver per LMS-CP-7151. The Center Director then decides whether to forward the waiver to Headquarters for approval.

3.2.2 The designated SMA Technical Authority is a mandatory approver for:

a. Tailoring of all procedural requirements when the software is safety-critical.
b. Tailoring of the following procedural requirements: SWE-022, SWE-023, SWE-032, SWE-131, SWE-134, SWE-141, and SWE-160.
c. Tailoring of procedural requirements that require an LMS waiver per paragraph 3.2.1.

3.2.3 The designated Engineering Technical Authority will assess the project's compliance matrix and tailoring by following NPR 7150.2B SWE-126. [SWE-126]

3.2.4 When tailoring requires an LMS waiver:

da. The NASA Software Lead shall process the request through LMS-CP-7151 Obtaining Waivers for Langley Management System (LMS) Requirements. [LSWE-047]
e. The Engineering and SMA Technical Authority chains shall be the Recommending Authorities in LMS-CP-7151 starting with the designated Engineering and SMA Technical Authorities for the software task. [LSWE-048]
f. For requirements listed in paragraph 3.2.1.a, the Approvers in LMS-CP-7151 shall be the Center Chief Engineer and the Director of the Safety and Mission Assurance Office. [LSWE-049]
g. For requirements listed in paragraph 3.2.1.b, the Approver in LMS-CP-7151 shall be the Center Director, and the Center Director shall forward approved requests, for approval, to the Headquarters Chief Engineer and the Headquarters Chief for Safety and Mission Assurance. [LSWE-050]
Appendix A. Supplemental Definitions

This appendix provides supplemental definitions for terms used in this LPR. The definitions in NPR 7150.2 Appendix A also apply to terms used in this LPR.

A.1 **Acceptance tests:** 1) Testing conducted to determine whether a system satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system. 2) Formal testing conducted to enable a user, customer, or other authorized entity to determine whether to accept a system or component.  [ISO/IEC/IEEE 24765:2017-09]

A.2 **Delivery:** Release of a system or component to its customer or intended user.  [ISO/IEC/IEEE 24765:2017-09]

A.3 **Demonstration:** Dynamic analysis technique that relies on observation of system or component behavior during execution, without need for post-execution analysis, to detect errors, violations of development standards, and other problems.  [ISO/IEC/IEEE 24765:2017-09]

A.4 **Developed software:** New software or software modifications developed by or for NASA.

A.5 **End-to-end tests:** The objective of end-to-end tests are to demonstrate interface compatibility and desired total functionality among different elements of a system, between systems, and systems as a whole. Interfaces include software/software, hardware/software, and system/system data exchanges. In addition, end-to-end tests include complete operational scenarios across system components to verify that performance requirements are met. End-to-end tests verify that the data flows throughout the multi-system environment are correct, the system provides the required functionality, and that the outputs at the eventual end points correspond to expected results.

A.6 **Function:** 1) Defined objective or characteristic action of a system or component. 2) Software module that performs a specific action, is invoked by the appearance of its name in an expression, receives input values, and returns a single value. 3) Transformation of inputs to outputs, by means of some mechanisms, and subject to certain controls, that is identified by a function name and modeled by a box.  [ISO/IEC/IEEE 24765:2017-09]

A.7 **Life cycle:** A partitioning of the life of a product or project into phases.  [CMMI–DEV V1.3]

A.8 **Maintenance:** 1) The process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment. 2) The process of retaining a hardware system or component in, or restoring it to, a state in which it can perform its required functions.  [ISO/IEC/IEEE 24765:2017-09]

A.9 **Non-developed software:** Existing software that is acquired or reused by the project. Non-developed software includes commercial-off-the-shelf (COTS) software, government-off-the-shelf (GOTS) software, modified-off-the-shelf (MOTS) software, heritage software, legacy software, reused software, and open-source software.

A.10 **Operational scenario:** Step-by-step description of how the proposed system should operate and interact with its users and its external interfaces (e.g., other systems). Scenarios should be described in a manner that will allow engineers to walk through them and gain an understanding of how all the various parts of the proposed system function and interact as well as validate that the system will satisfy the user’s needs and expectations. Operational scenarios should be described for all operational modes, mission phases (e.g., installation, startup, typical examples of normal and contingency operations, shutdown, maintenance, and safing), and critical sequences of activities for all classes of users identified. Each scenario should include events, actions, stimuli, information, and interactions as appropriate to provide a comprehensive understanding of the operational aspects of the system.  [Based upon IEEE 1362-1998] Operational scenarios should span all the following items (during nominal, off-nominal, and stressful conditions) that could occur during operations: mission phase, mode, and state transitions; first-time events; operational performance limits; fault protection routines; failure detection, isolation, and recovery logic; operational responses to transient or off-nominal sensor signals; ground-to-spacecraft uplink and downlink.

A.11 **Procedural requirement:** A requirement that governs the software engineering activities of a project or organization. For LaRC, procedural requirements are defined in the NASA Online Directives.
Information System (NODIS) and the Langley Management System (LMS). Procedural requirements are also found in NASA standards when a governing document in NODIS invokes the standard.

A.12 **Product Analysis**: Process of evaluating a product by manual or automated means to determine if the product has certain characteristics [ISO/IEC/IEEE 24765:2017-09]

A.13 **Project Portfolio**: A set of software tasks within the same domain that an organization manages using a common software engineering process.

A.14 **Qualification testing**: 1) Testing, conducted by the developer and witnessed by the acquirer (as appropriate), to demonstrate that a software product meets its specifications and is ready for use in its target environment or integration with its containing system. 2) Testing conducted to determine whether a system or component is suitable for operational use. [ISO/IEC/IEEE 24765:2017-09]

A.15 **Regression testing**: 1) Selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements. [ISO/IEC/IEEE 24765:2017-09]

A.16 **Release**: 1) Particular version of a configuration item that is made available for a specific purpose (for example, test release). 2) Collection of new or changed configuration items that are tested and introduced into a live environment together. 3) Collection of one or more new or changed configuration items deployed into the live environment as a result of one or more changes. 4) Software version that is made formally available to a wider community. 5) Delivered version of an application which includes all or part of an application. 6) Set of grouped change requests, established in the Application Change Management Process, which are designed, developed, tested, and deployed as a cohesive whole. [ISO/IEC/IEEE 24765:2017-09]

A.17 **Software item**: 1) Aggregation of software, such as a computer program or database, that satisfies an end use function and is designated for purposes of specification, qualification testing, interfacing, configuration management, or other purposes. 2) Source code, object code, control code, control data, or a collection of these items. 3) Identifiable part of a software product. [ISO/IEC/IEEE 24765:2017-09]

A.18 **Software peer review or inspection**: 1) Visual examination of a software product to detect and identify software anomalies, including errors and deviations from standards and specifications. 2) Static analysis technique that relies on visual examination of development products to detect errors, violations of development standards, and other problems. 3) Examining or measuring to verify whether an activity, component, product, result, or service conforms to specified requirements. [ISO/IEC/IEEE 24765:2017-09] Guidelines for software peer reviews/inspections are contained in NASA-STD-2202-93, Software Formal Inspection Standard. [NPR 7150.2B: Appendix A]

A.19 **Software product**: 1) Set of computer programs, procedures, and possibly associated documentation and data. 2) Any of the individual items in (1). 3) Complete set of software designed for delivery to a software consumer or end-user, which can include computer programs, procedures and associated documentation and data. 4) Set of computer programs, procedures, database and other data structure descriptions, and associated documentation. [ISO/IEC/IEEE 24765:2017-09]

A.20 **Software unit**: 1) Atomic-level software component of the software architecture that can be subjected to standalone testing. 2) Separately compilable piece of code. [ISO/IEC/IEEE 24765:2017-09]

A.21 **Stakeholders**: Individuals that are affected by or in some way accountable for the outcome of the project (may include project members, suppliers, customer/acquirer [SWE102], end users, and others). [CMMI-DEV V1.3] Stakeholders also include the project manager, senior management, subsystem leads, and software quality assurance.

A.22 **Test**: 1) Activity in which a system or component is executed under specified conditions, the results are observed or recorded, and an evaluation is made of some aspect of the system or component. 2) To conduct an activity as in (1). 3) Set of one or more test cases and procedures. [ISO/IEC/IEEE 24765:2017-09]

A.23 **Verification methods** (also known as qualification provisions or qualification methods) may include: demonstration, test, product analysis, or software peer review or inspection.

Page 16 of 53

Verify the correct revision before use by checking the LMS Web site.
### Appendix B. Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASIC</td>
<td>Application Specific Integrated Circuits</td>
</tr>
<tr>
<td>CMMI®</td>
<td>Capability Maturity Model Integration</td>
</tr>
<tr>
<td>CMMI-DEV</td>
<td>Capability Maturity Model® Integration® (CMMI®) for Development</td>
</tr>
<tr>
<td>CMU</td>
<td>Carnegie Mellon University</td>
</tr>
<tr>
<td>COTS</td>
<td>Commercial-Off-The-Shelf</td>
</tr>
<tr>
<td>CP</td>
<td>Center Procedure</td>
</tr>
<tr>
<td>CPLD</td>
<td>Complex Programmable Logic Device</td>
</tr>
<tr>
<td>FAR</td>
<td>Federal Acquisition Regulation</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Arrays</td>
</tr>
<tr>
<td>GOTS</td>
<td>Government-Off-The-Shelf</td>
</tr>
<tr>
<td>HQ</td>
<td>Headquarters</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineers</td>
</tr>
<tr>
<td>IV&amp;V</td>
<td>Independent Verification and Validation</td>
</tr>
<tr>
<td>LaRC</td>
<td>Langley Research Center</td>
</tr>
<tr>
<td>LAPD</td>
<td>Langley Policy Directive</td>
</tr>
<tr>
<td>LMS</td>
<td>Langley Management System</td>
</tr>
<tr>
<td>LMS CP</td>
<td>Langley Management System Center Procedure</td>
</tr>
<tr>
<td>LPR</td>
<td>Langley Procedural Requirements</td>
</tr>
<tr>
<td>MAB</td>
<td>Mission Assurance Branch</td>
</tr>
<tr>
<td>MOTS</td>
<td>Modified-Off-The-Shelf</td>
</tr>
<tr>
<td>NA</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>NASA</td>
<td>National Aeronautics and Space Administration</td>
</tr>
<tr>
<td>NODIS</td>
<td>NASA Online Directives Information System</td>
</tr>
<tr>
<td>NPD</td>
<td>NASA Policy Directive</td>
</tr>
<tr>
<td>NPR</td>
<td>NASA Procedural Requirements</td>
</tr>
<tr>
<td>OCE</td>
<td>Office of Chief Engineer</td>
</tr>
<tr>
<td>RFP</td>
<td>Request for Proposal</td>
</tr>
<tr>
<td>SEI</td>
<td>Software Engineering Institute</td>
</tr>
<tr>
<td>SEPG</td>
<td>Software Engineering Process Group</td>
</tr>
<tr>
<td>SMA</td>
<td>Safety and Mission Assurance</td>
</tr>
<tr>
<td>SMP</td>
<td>Software Management Plan</td>
</tr>
<tr>
<td>SoC</td>
<td>System on Chip</td>
</tr>
<tr>
<td>SOW</td>
<td>Statement of Work</td>
</tr>
<tr>
<td>SWE</td>
<td>Software Engineering</td>
</tr>
<tr>
<td>SWG</td>
<td>Software Working Group</td>
</tr>
<tr>
<td>TA</td>
<td>Technical Authority</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>WBS</td>
<td>Work Breakdown Structure</td>
</tr>
</tbody>
</table>

Verify the correct revision before use by checking the LMS Web site.
Appendix C. References

Note: NPDs and NPRs are found in the NASA Online Directives Information System (NODIS) at: 
http://nodis3.gsfc.nasa.gov/  Standards are found on the NASA Technical Standards Program Web site at:  https://standards.nasa.gov/
Note: References that are also listed as applicable documents in section P.4 are not repeated here.

d. NASA-HDBK 8739.23, NASA Complex Electronics Handbook for Assurance Professionals,  
e. CMU/SEI-2010-TR-033, CMMI® for Development, Version 1.3,  
http://www.sei.cmu.edu/library/abstracts/reports/10tr033.cfm.
f. LaRC Measure Definitions,  https://sw-eng.larc.nasa.gov/supporting-products/
Appendix D. LaRC Guidance on Compliance Matrices and Tailoring

The Compliance Matrix documents the intent of a project or portfolio of projects to comply with the requirements of NPR 7150.2 and documents any tailoring of requirements. This appendix presents LaRC guidance on compliance matrix content and guidance on compliance determination.

D.1 Minimum Content

This section defines the minimum content required of a compliance matrix. Projects and organizations are free to define the format of the compliance matrix. For convenience, the Langley Software Engineering Process Group provides templates that meet the requirements of this LPR and are pre-populated with the tailoring recommended in this appendix. Those templates are hosted on the Software Process Improvement Initiative web-site at http://sw-eng.larc.nasa.gov/.

A compliance matrix will contain the following content at a minimum:

- The unique requirement identifier from NPR 7150.2 or LPR 7150.2 for each requirement mapped to the software class per NPR 7150.2B Appendix C as augmented by section 2.3.
- The modifications to requirements mapping in section 2.3.
- The responsible party (i.e., the organization or individual) responsible for the requirement.
- The level to which the project plans to comply with the requirement. The compliance levels are:
  - FC – full compliance
  - T – tailored. Used to tailor down or tailor out a requirement.
  - NA – not applicable. Used when the project does not meet the restricted applicability of a requirement (see Appendix D.2).

See section 3.1 for additional information on the compliance levels. Also Appendix D.4 lists requirements for which projects become fully compliant by adhering to this LPR.

- For each requirement with an entry of T (tailored), include the following information:
  - A description of the requirement tailoring for the project or project portfolio.
  - Tailoring rationale to justify tailoring of the requirement.

- For each requirement with an entry of NA (not applicable), identify the restriction that does not apply (e.g., 'not a spaceflight project') as justification.

- Signatories. Signatures can be electronic. The required signatories are listed below. Projects and organizations are free to add signatories:
  - NASA Software Lead (approver)
  - NASA Software Lead's Supervisor (approver)
  - Center Software Assurance Manager (concurrence – performs software assurance on the compliance matrix)
  - Engineering Technical Authority (approver if compliance matrix includes tailoring).
  - SMA Technical Authority (approver per section 3.2 if compliance matrix includes tailoring).

D.2 Restricted Applicability

This section summarizes the NPR 7150.2 requirements that have restricted applicability as documented in NPR 7150.2. These requirements can be marked NA without requiring approval of the technical authority. This NA designation is not a generic 'not applicable' designation as it is used only for NPR 7150.2 requirements with inherent restricted applicability. For other situations where a NPR 7150.2 requirement is not applicable due to the unique circumstances of a project or project portfolio, removing the requirement is a 'tailor out' under the 'T' designation. SWE-126 requires such 'not applicable' determinations to be reviewed and confirmed by the technical authority.

Supplier Agreements Only

The following requirements apply only to projects that use supplier agreements for some or all of the software products or software engineering activities:

SWE-035  SWE-037  SWE-038  SWE-039  SWE-040  SWE-041  SWE-042

Verify the correct revision before use by checking the LMS Web site.
## Spaceflight Only
The following requirements apply only to projects developing ground or flight software for space missions:

<table>
<thead>
<tr>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWE-070</td>
</tr>
<tr>
<td>SWE-143</td>
</tr>
<tr>
<td>SWE-154</td>
</tr>
<tr>
<td>SWE-155</td>
</tr>
<tr>
<td>SWE-156</td>
</tr>
<tr>
<td>SWE-157</td>
</tr>
<tr>
<td>SWE-158</td>
</tr>
<tr>
<td>SWE-159</td>
</tr>
</tbody>
</table>

SWE-143 has additional restrictions stated in the requirement that are justification for an N/A when those restrictions do not apply to the spaceflight project.

## Safety-Critical Only
The following requirements apply only when developing software of the specified class and the software is safety-critical.

<table>
<thead>
<tr>
<th>Class</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class C</td>
<td>SWE-023</td>
</tr>
<tr>
<td></td>
<td>SWE-134</td>
</tr>
<tr>
<td>Class D</td>
<td>SWE-017</td>
</tr>
<tr>
<td></td>
<td>SWE-023</td>
</tr>
<tr>
<td></td>
<td>SWE-027</td>
</tr>
<tr>
<td></td>
<td>SWE-040</td>
</tr>
<tr>
<td></td>
<td>SWE-041</td>
</tr>
<tr>
<td></td>
<td>SWE-042</td>
</tr>
<tr>
<td></td>
<td>SWE-043</td>
</tr>
<tr>
<td></td>
<td>SWE-045</td>
</tr>
<tr>
<td></td>
<td>SWE-047</td>
</tr>
<tr>
<td></td>
<td>SWE-051</td>
</tr>
<tr>
<td></td>
<td>SWE-052</td>
</tr>
<tr>
<td></td>
<td>SWE-054</td>
</tr>
<tr>
<td></td>
<td>SWE-055</td>
</tr>
<tr>
<td></td>
<td>SWE-058</td>
</tr>
<tr>
<td></td>
<td>SWE-059</td>
</tr>
<tr>
<td></td>
<td>SWE-061</td>
</tr>
<tr>
<td></td>
<td>SWE-064</td>
</tr>
<tr>
<td></td>
<td>SWE-067</td>
</tr>
<tr>
<td></td>
<td>SWE-073</td>
</tr>
<tr>
<td></td>
<td>SWE-079</td>
</tr>
<tr>
<td></td>
<td>SWE-082</td>
</tr>
<tr>
<td></td>
<td>SWE-083</td>
</tr>
<tr>
<td></td>
<td>SWE-084</td>
</tr>
<tr>
<td></td>
<td>SWE-087</td>
</tr>
<tr>
<td></td>
<td>SWE-088</td>
</tr>
<tr>
<td></td>
<td>SWE-089</td>
</tr>
<tr>
<td></td>
<td>SWE-090</td>
</tr>
<tr>
<td></td>
<td>SWE-093</td>
</tr>
<tr>
<td></td>
<td>SWE-094</td>
</tr>
<tr>
<td></td>
<td>SWE-134</td>
</tr>
<tr>
<td></td>
<td>SWE-135</td>
</tr>
<tr>
<td></td>
<td>SWE-136</td>
</tr>
<tr>
<td></td>
<td>SWE-146</td>
</tr>
<tr>
<td></td>
<td>SWE-149</td>
</tr>
</tbody>
</table>

## Independent Verification and Validation (IV&V) Only
SWE-131 is not applicable if the project is not selected for software IV&V.
### D.3 Intent Clarifications for Select NPR 7150.2 Requirements by Class

Table D.1 presents select restatements of NPR 7150.2 requirements that are appropriately scoped to the software class. These restatements are intended to clarify how lower classes are expected to meet the intent of the selected requirements.

<table>
<thead>
<tr>
<th>NPR SWE #</th>
<th>Requirement Text</th>
<th>Software Class</th>
<th>Modified Text for Class</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>The project manager shall develop, maintain, and execute software plans that cover the entire software life cycle and, as a minimum, address the requirements of this directive with approved tailoring.</td>
<td>Class E</td>
<td>The project manager shall develop, maintain, and execute a software management plan.</td>
<td>Class E projects are low cost and low risk. Therefore, a Class E project can get by with a minimal software management plan that a) identifies the scope, planned effort and staffing, and planned completion date for the task and b) outlines the software development process.</td>
</tr>
</tbody>
</table>
| 16       | The project manager shall document and maintain a software schedule that satisfies the following conditions:  
  a. Coordinates with the overall project schedule.  
  b. Documents the interactions of milestones and deliverables between software, hardware, operations, and the rest of the system.  
  c. Reflects the critical path for the software development activities.  
  d. Adhere to the guidance provided in NASA/SP-2010-3403, NASA Scheduling Management Handbook. | Class C        | The project manager shall document and maintain a software schedule that satisfies the following conditions:  
  a. Coordinates with the overall project schedule.  
  b. Documents the interactions of milestones and deliverables between software, hardware, operations, and the rest of the system.  
  c. Reflects the critical path for the software development activities.  
  < item (d) deleted >                                                                 | NASA/SP-2010-3403 provides scheduling guidance for NPR 7120.5, 7120.7, and 7120.8. The first two documents set a $20 million dollar floor for applying the schedule techniques covered by this NASA/SP; and 7120.8 leaves this decision up to the mission directorates. Furthermore, LaRC’s strategic framework places responsibility for scheduling management in the Product Unit Directorates (PUDs) where the Program and Project Managers reside. Software projects normally reside in the Core Resource Unit Directorates (CRUDs) and must follow the scheduling direction of the PUDs. Therefore, applicability of NASA/SP-2010-3403 is a Program Level 1 or Project Level 2 decision that applies to the whole program or project and flows down to the software project; this guidance cannot be implemented by software in a vacuum. |
<table>
<thead>
<tr>
<th>NPR SWE #</th>
<th>Requirement Text</th>
<th>Software Class</th>
<th>Modified Text for Class</th>
<th>Rationale</th>
</tr>
</thead>
</table>
|          |                 | Class D        | The project manager shall document and maintain a software schedule that satisfies the following conditions:  
  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.  
  < items (c) and (d) deleted. > | Class D projects may manage to a milestone-based or iteration-based schedule (e.g. sprints), which does not generate a critical path.  
See rationale for deleted item (d) under Class C entry above. |
| 22       | The project manager shall plan and implement software assurance per NASA-STD-8739.8. | Class D | The project manager shall:  
  a. Submit classification and safety-critical determination for independent assessment per LMS-CP-4754  
  b. Submit a completed compliance matrix to the NASA Software Assurance Manager for concurrence per LSWE-025  
  c. Shall cooperate with spot checks initiated by Mission Assurance Branch | NASA-STD-8739.8 allows substantially tailoring of its requirements for Class D software. LaRC Engineering and LaRC Mission and Safety Assurance have agreed that these are the minimal software assurance activities for Class D projects. |
| 35       | The project manager shall establish a procedure for software supplier selection, including proposal evaluation criteria. | A through E | The project shall contact the LaRC Office of Procurement when acquiring software and shall participate in the procurement process.  
[Note: LPR 7150.2 levies this requirement on class E – See section 2.3] | Supplier selection procedure is governed by laws and regulations outside of NPR 7150.2, and the Center has defined procedures that address all procurement requirements. These procedures apply regardless of software class or safety-criticality. |
<table>
<thead>
<tr>
<th>NPR SWE #</th>
<th>Requirement Text</th>
<th>Software Class</th>
<th>Modified Text for Class</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>The project manager shall establish, capture, record, approve, and maintain software requirements, including the software quality requirements, as part of the technical specification.</td>
<td>Class E</td>
<td>The project manager shall record and maintain a description of the software with sufficient detail to inform implementation.</td>
<td>Class E software is low cost, low risk, and often exploratory. Class E projects have freedom to informally develop and maintain a description of the software's form and function. This description may take the form of goals, use cases, concepts of operations, features, user stories, tasks, or requirements.</td>
</tr>
<tr>
<td>58</td>
<td>The project manager shall develop, record, and maintain a design based on the software architectural design that describes the lower-level units so that they can be coded, compiled, and tested.</td>
<td>Class D and safety-critical</td>
<td>The project manager shall develop, record, and maintain a design that describes the lower-level units so that they can be coded, compiled, and tested.</td>
<td>The requirement for a software architecture description (SWE-057) is not mapped to Class D.</td>
</tr>
</tbody>
</table>
| 59      | The project manager shall perform, record, and maintain bidirectional traceability between the following:  
  a. Software requirements and software architecture.  
  b. Software architecture and software design.  
  c. Software requirements and software design. | Class C                | The project manager shall perform, record, and maintain bidirectional traceability between software requirements and the software design.                  | For software that is not safety-critical and low cost, it is sufficient to perform bidirectional traceability between software requirements and the higher-level requirements, the detailed design, and the test procedures. These three traces provide minimal verification that the software requirements, software design, and software implementation do address all user or system requirements. Though this reduces opportunities for earlier detection of inconsistencies among software requirements and software products, the cost of additional traces exceeds the savings of early detection given the low-complexity typical of the software described. |
<p>|         |                                                                                                                                                                                                                     | Class D and safety-critical | The project manager shall perform, record, and maintain bidirectional traceability between software requirements and the software design.                  | The requirement for a software architecture description (SWE-057) is not mapped to Class D.                                                                                                               |</p>
<table>
<thead>
<tr>
<th>NPR SWE #</th>
<th>Requirement Text</th>
<th>Software Class</th>
<th>Modified Text for Class</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>The project manager shall implement the software design into software code.</td>
<td>Class D</td>
<td>The project manager shall implement the software requirements into software code.</td>
<td>The requirements for developing a software architecture or design (SWE-056, SWE-057, and SWE-058) are not mapped to Class D if not safety-critical.</td>
</tr>
<tr>
<td>64</td>
<td>The project manager shall perform, record, and maintain bidirectional traceability from software design to the software code.</td>
<td>Class C</td>
<td>When the cost of the NASA system exceeds $20 million or the software is safety-critical, the project manager shall perform, record, and maintain bidirectional traceability from software design to the software code.</td>
<td>For software that is not safety-critical and low cost, it is sufficient to perform bidirectional traceability between software requirements and the higher-level requirements, the detailed design, and the test procedures. These three traces provide minimal verification that the software requirements, software design, and software implementation do address all user or system requirements. Though this reduces opportunities for earlier detection of inconsistencies among software requirements and software products, the cost of additional traces exceeds the savings of early detection given the low-complexity typical of the software described.</td>
</tr>
<tr>
<td>65</td>
<td>The project manager shall establish and maintain:</td>
<td>Class D</td>
<td>The project manager shall establish and maintain:</td>
<td>Class D software that is not safety-critical does not typically require a certification process (e.g. safety certification or certification for flight) or formal review (e.g. Operational Readiness Review) to place it in operation and, therefore, does not need the level of formal test documentation required for such certifications or formal reviews.</td>
</tr>
<tr>
<td></td>
<td>a. Software test plan(s).</td>
<td></td>
<td>a. A software test plan</td>
<td></td>
</tr>
<tr>
<td></td>
<td>b. Software test procedure(s).</td>
<td></td>
<td>b. Descriptions of the test cases</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c. Software test report(s).</td>
<td></td>
<td>c. Software test status (e.g., pass-fail).</td>
<td></td>
</tr>
<tr>
<td>NPR SWE #</td>
<td>Requirement Text</td>
<td>Software Class</td>
<td>Modified Text for Class</td>
<td>Rationale</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------</td>
<td>------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>71</td>
<td>The project manager shall update software test plan(s) and software test procedure(s) to be consistent with software requirements.</td>
<td>Class D</td>
<td>The project manager shall update the software test plan and software test cases to be consistent with software requirements.</td>
<td>Class D software that is not safety-critical does not typically require a certification process (e.g. safety certification or certification for flight) or formal review (e.g. Operational Readiness Review) to place it in operation and, therefore, does not need the level of formal test documentation required for such certifications or formal reviews.</td>
</tr>
<tr>
<td>77</td>
<td>The project manager shall complete and deliver the software product to the customer with appropriate records, including as-built records, to support the operations and maintenance phase of the software’s life cycle.</td>
<td>Class E</td>
<td>The project manager shall complete and deliver the software product to the customer and fulfill commitments made to the customer for software documentation or data.</td>
<td>NPR 7150.2B requirements to produce artifacts (e.g. design descriptions, test procedures) that would fall under the ‘appropriate records’ clause are not mapped to Class E. Moreover, NPR 7150.2B does not require class E projects to define a life cycle or plan for operations and maintenance.</td>
</tr>
</tbody>
</table>
| 88       | The project manager shall, for each planned software peer review or software inspection:  
  a. Use a checklist or formal reading technique to evaluate the work products.  
  b. Use established readiness and completion criteria.  
  c. Track actions identified in the reviews until they are resolved.  
  d. Identify required participants. | Class C, Class D + Safety-Critical | The project manager shall, for each planned software peer review or software inspection:  
  a. Use established readiness and completion criteria.  
  b. Track actions identified in the reviews until they are resolved.  
  c. Identify required participants.  
  < original item (a) deleted >                                                                 | Based on the experience of LaRC software practitioners, peer reviews of Class C and Class D software, regardless of safety-critical determination, can be effective without using checklists or formal reading techniques. The experience of the reviewers is often more important to a successful review than the review method and aids used. |
<table>
<thead>
<tr>
<th>NPR SWE #</th>
<th>Requirement Text</th>
<th>Software Class</th>
<th>Modified Text for Class</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>89</td>
<td>The project manager shall, for each planned software peer review or software inspection, record basic measurements.</td>
<td>Class C, Class D + Safety-Critical</td>
<td>When the cost of the NASA system exceeds $20 million, the project manager shall, for each planned software peer review or software inspection, record basic measurements</td>
<td>Basic software measurements are used to quantitatively manage and optimize the peer review process. The value of the measurements, therefore, is related to the number of reviews that the project will conduct. Low-cost projects are unlikely to hold a sufficient number of reviews to justify the cost of measuring and analyzing those reviews.</td>
</tr>
<tr>
<td>146</td>
<td>The project manager shall define the approach to the automatic generation of software source code including:</td>
<td>Class C</td>
<td>The project manager shall define the approach to the automatic generation of software source code including:</td>
<td>The cost associated with the monitoring of auto-generated software use is often not justified in projects whose software is not safety-critical.</td>
</tr>
</tbody>
</table>
The project manager's software cost estimate(s) shall satisfy the following conditions:

a. Covers the entire software life cycle.
b. Is based on selected project attributes.
c. Is based on the cost implications of the technology to be used and the required maturation of that technology.
d. Incorporates risk and uncertainty.
e. Includes the cost for software assurance support.
f. Includes other direct costs.

Class D is not required to perform risk management and, therefore, cannot be expected to identify risks that impact the cost estimate. LaRC tailoring of NASA-STD-8739.8 for Class D leaves no costs to be incurred by the projects.

### D.4 Requirements with Fully Compliant Implementation in LPR 7150.2

This section identifies the NPR 7150.2 requirements that are mapped to projects and that have a fully compliant implementation in this LPR. Each project should duplicate these entries in their compliance matrix.

#### Table D.2 NPR 7150.2 Requirements Implemented in LPR 7150.2

<table>
<thead>
<tr>
<th>NPR SWE #</th>
<th>Requirement Text</th>
<th>Software Class</th>
<th>Responsible Party</th>
<th>Compliance</th>
<th>LPR Paragraph</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>The project manager shall classify each system and subsystem containing software in accordance with the highest applicable software classification definitions for Classes A, B, C, D, E, F, G, and H software in Appendix D.</td>
<td>A, B, C, D, E</td>
<td>NASA Software Lead</td>
<td>FC</td>
<td>2.2.1</td>
</tr>
<tr>
<td>21</td>
<td>If a system or subsystem evolves to a higher or lower software classification as defined in Appendix D, or there is a change in the safety criticality of the software, then the project manager shall update their plan to fulfill the applicable requirements per the Requirements Mapping and Compliance Matrix in Appendix C and any approved tailoring.</td>
<td>A, B, C, D, E</td>
<td>Project</td>
<td>FC</td>
<td>2.2.5</td>
</tr>
<tr>
<td>33</td>
<td>The project manager shall assess options for software acquisition versus development.</td>
<td>A, B, C, D, E</td>
<td>NASA Software Lead</td>
<td>FC</td>
<td>2.6.1</td>
</tr>
<tr>
<td>35</td>
<td>The project manager shall establish a procedure for software supplier selection, including proposal evaluation criteria.</td>
<td>A, B, C, D, E</td>
<td>NASA Software Lead</td>
<td>FC</td>
<td>2.6.2</td>
</tr>
<tr>
<td>36</td>
<td>The project manager shall determine which software processes, software documents, electronic products, software activities, and tasks are required for the project and software suppliers.</td>
<td>A, B, C, D, E</td>
<td>NASA Software Lead</td>
<td>FC</td>
<td>2.6.5</td>
</tr>
<tr>
<td>125</td>
<td>Each project manager with software components shall maintain a compliance matrix or multiple compliance matrices against requirements in this NPR, including those delegated to other parties or accomplished by contract vehicles or Space Act Agreements.</td>
<td>A, B, C, D, E</td>
<td>NASA Software Lead</td>
<td>FC</td>
<td>2.4.1</td>
</tr>
<tr>
<td>132</td>
<td>The project's software assurance manager shall perform an independent classification assessment.</td>
<td>A, B, C, D, E</td>
<td>Center Software Assurance Manager</td>
<td>FC</td>
<td>2.2.4</td>
</tr>
<tr>
<td>133</td>
<td>The project manager, in conjunction with the Safety and Mission Assurance organization, shall determine the software safety criticality in accordance with NASA-STD-8719.13.</td>
<td>A, B, C, D, E</td>
<td>NASA Software Lead and Center Software Assurance Manager</td>
<td>FC</td>
<td>2.2.3, 2.2.4</td>
</tr>
<tr>
<td>139</td>
<td>The projects shall comply with the requirements in this NPR that are marked with a &quot;project&quot; responsibility and an &quot;X&quot; in Appendix D consistent with their software classification.</td>
<td>A, B, C, D, E</td>
<td>Project</td>
<td>FC</td>
<td>2.1.1</td>
</tr>
<tr>
<td>145</td>
<td>When the compliance matrix is used to waive/deviate from applicable &quot;X&quot; requirement(s), the designated Technical Authorities shall indicate their approval by signature(s) in the compliance matrix itself.</td>
<td>A, B, C, D, E</td>
<td>Designated Technical Authorities</td>
<td>FC</td>
<td>2.4.3, 3</td>
</tr>
</tbody>
</table>
Appendix E. Recommended Document Contents
Per NPR 7150.2B SWE-153, "The designated Engineering Technical Authority(s) shall define the content requirements for software documents or records." This appendix lists recommended document contents for Software Classes A through E, and addresses documents that Langley Research Center expects the Engineering TA to define. For convenience, required documents that fall under the authority of the Safety and Mission Assurance (SMA) TA are also referenced here. This appendix does not list all the documents that a project is expected to create in order to fulfill NPR 7150.2B requirements. Projects are free to define the content of additional documents. For example, this appendix does not define content for a schedule. The content of the schedule will depend on the practices and tools that the project selects for project planning and tracking.

Instructions for Technical Authorities
Per section 2 of this LPR, the designated Engineering TA may levy this appendix on a project or project portfolio. If the software is safety-critical, both the Engineering TA and the SMA TA must agree to levy this appendix. The Engineering TA is also free to define or approve alternative documentation requirements, and those requirements are free to deviate from the content of this appendix. For example, in lieu of levying this appendix, the Engineering TA can approve the use of established documentation standards within an organization for that organization's project portfolio.

Instructions for Projects
A project is required to conform to the content of this appendix only when directed by its designated Engineering TA (see section 2). When so directed, the project must obtain written approval of the Engineering TA and, if safety-critical, the SMA TA to tailor down the content of this appendix. However, no approval is necessary to augment the content of this appendix.

The project first uses Table E.1 and Table E.2 to identify the documents that it will develop. Table E.1 lists the documents under the authority of the Engineering TA. Table E.2 lists the documents under the authority of the SMA TA. Table E.2 is provided for convenience of reference, and tailoring of documentation in this table requires approval of the designated SMA TA. Below each document, in parenthesis, is the section of this LPR or the NASA standard that defines the content for that document. The 'SWE' column provides the NPR 7150.2 requirement number(s), for which the document is a typical work product. The remaining columns identify the applicability of the document to each software class and use the following markers:

- <blank> The document is optional for the software class.
- X The document is required for the software class.
- S/C The document is required for the software class when the software is safety-critical; otherwise it is optional.
- Note^n The footnote describes tailoring guidance or restrictions for the software class.

Table E.1 Documents under the Engineering Technical Authority by Class

<table>
<thead>
<tr>
<th>Documentation / Record</th>
<th>SWE</th>
<th>Class A</th>
<th>Class B</th>
<th>Class C</th>
<th>Class D</th>
<th>Class E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Compliance Matrix (Appendix D)</td>
<td>SWE-125</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Software Management Plan (Appendix E.1)</td>
<td>SWE-013</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Software Configuration Management Plan (Appendix E.2)</td>
<td>SWE-079</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>S/C</td>
<td></td>
</tr>
<tr>
<td>Software Test Plan (Appendix E.3)</td>
<td>SWE-065</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Software Maintenance Plan (Appendix E.4)</td>
<td>SWE-075</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Documentation / Record</td>
<td>SWE</td>
<td>Class A</td>
<td>Class B</td>
<td>Class C</td>
<td>Class D</td>
<td>Class E</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>Software Requirements Specification (Appendix E.5)</td>
<td>SWE-050</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Note 1</td>
</tr>
<tr>
<td>Software Architecture Description (Appendix E.6)</td>
<td>SWE-056</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>S/C</td>
<td></td>
</tr>
<tr>
<td>Software Design Description (Appendix E.7)</td>
<td>SWE-056</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>S/C</td>
<td></td>
</tr>
<tr>
<td>Software Test Procedures (Appendix E.8)</td>
<td>SWE-065</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>S/C</td>
<td></td>
</tr>
<tr>
<td>Software Test Report (Appendix E.9)</td>
<td>SWE-065</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>S/C</td>
<td></td>
</tr>
<tr>
<td>Software Change Request (Appendix E.10)</td>
<td>SWE-080</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>S/C</td>
<td></td>
</tr>
<tr>
<td>Software Problem Report (Appendix E.11)</td>
<td>SWE-069</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Software Version Description (Appendix E.12)</td>
<td>SWE-081</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Software User’s Manual (Appendix E.13)</td>
<td>SWE-077</td>
<td>X</td>
<td>X</td>
<td>S/C</td>
<td>S/C</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Documentation / Record</th>
<th>SWE</th>
<th>Class A</th>
<th>Class B</th>
<th>Class C</th>
<th>Class D</th>
<th>Class E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Assurance Classification Report (LMS-CP-4754)</td>
<td>SWE-132</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Software Assurance Plan (NASA-STD-8739.8)</td>
<td>SWE-022</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>S/C</td>
<td>Note 4</td>
</tr>
</tbody>
</table>

The project next develops the identified documents according to their referenced content definitions. For documents defined in this appendix, each document is defined using a list of content items. Each content item is presented in three parts:

- a short name for the content item (underlined),
- the applicable software classes (in braces) with the added designation of 'S/C' to identify applicability to 'safety-critical' software regardless of software class, and
- a short description of the content item.

The short name and applicable software class are normative; i.e., the project cannot tailor these without TA approval. The description is informative. The project is not required to adhere to the description as stated but should meet the intent of the description at a level appropriate for the software class and safety-critical designation. In fact, the description is written to cover the breadth of applicable classes and may contain elements that NPR 7150.2B does not impose on lower classes.

---

1 For Class E, NPR 7150.2B does not require validation, bidirectional traceability, or change tracking of software requirements; therefore, software requirements can be expressed as an informal description of the software's intended purpose or planned features.
2 Appendix D substitutes less formal documentation for software that is Class D and not safety-critical.
3 A project developing Class D and not safety-critical software is required to 'track and evaluate changes to software products' (SWE-080), but is free to define its software change documentation.
4 Software assurance for software that is Class D and not safety-critical has been integrated into LPR 7150.2 and LMS-CP-4754 Software Assurance (SA) for Development and Acquisition.
Each document is tailored to the software class in two ways. The first is through the use of the recommended software class for each content item in the braces. The second is by the level of rigor and detail, with which each applicable content item is addressed. The level of rigor should satisfy all the related NPR 7150.2B requirements mapped to the software class. For example, one should expect the description of the verification and validation process in a Class A project to be more rigorous and lengthy than the same process description for a Class D project, in part because more NPR 7150.2B verification and validation requirements are mapped to class A than to class D.

No form or format should be implied from the content definitions in this appendix. Documents may take the form of flat files (e.g. MS Word), hyperlinked documents (e.g. wiki), databases, models, etc. However, the project should take into account the needs of a document's stakeholders and may need to convert the content into different forms. Likewise, the project may combine or further subdivide documents without approval of the Engineering TA as long as all the applicable content of the documents are addressed.
E.1 **Software Management Plan (SMP)**

The Software Management Plan should include:

a. **Scope.** {A-E} The scope identifies the project and the software products and services within the scope of the project. The scope may also address the following topics: relationship of the software project to higher-level projects, the system context for the software products, a summarized concept of operation, relationship of the plan to other plans, and software products or services not within scope.

b. **Software classification and safety criticality.** {A-E} Lists each of the systems and subsystems containing software that are within scope and identify the software classification and safety-critical determination of those system and subsystems.

c. **Project organization.** {A-D} Project organizational structure showing authority and responsibility of each organizational unit, including contractual suppliers and external organizations (e.g., universities, other government organizations, and industry partners).

d. **Staffing.** {A-E} Identify the individuals assigned to the project and their time commitment. Large projects may use a workforce management solution for staffing and may document the workforce management process in this plan.

e. **Budget.** {A-D} Describe the process for estimating, documenting, and tracking software effort and costs. Reference a higher-level plan if the budget for software is managed at a higher level.

f. **Schedule.** {A-D} Describe the process for establishing, maintaining, and tracking the software schedule. Reference a higher-level plan if the budget for software is managed at a higher level. Higher class projects may further address work breakdown structures and resource scheduling. For projects performing only software maintenance, describe how approved change requests are scheduled for implementation and tracked to completion.

g. **Reviews.** {A-D}. Identify the reviews that are conducted with stakeholders.

h. **Contractor management.** {A-D} Describe the process for maintaining insight and oversight of software products or services performed under contract.

i. **Life cycle.** {A-D} Document the software-development life cycle including the transition criteria for life cycle phases.

j. **Requirements management.** {A-D} Describe the process to develop and maintain the software requirements, establish and maintain requirements traceability, and track requirements changes.

k. **Design process.** {A-C, S/C} Describe the process for developing and maintaining the software design.

l. **Implementation process.** {A-D} Describe the process, techniques, or tools for software construction. This section should also address, as applicable, how the project will meet NPR 7150.2B requirements for off-the-shelf software, open source software, and auto-generated software.

m. **Verification and validation.** {A-E} Describe the verification and validation process. This item should address the verification and validation methods and techniques (e.g. peer review, test, static analysis), how those methods and techniques are integrated into the software lifecycle, and how the project documents and tracks defects to closure.

n. **Risk management.** {A-C} Define how the project will identify, analyze, plan, track, control, communicate, and document software risks.

o. **Software metrics.** {A-C, S/C} Describe the process to establish, record, maintain, report, and utilize software management and technical measurements.

p. **Intellectual property rights.** {A-E} Identify the intellectual property rights that NASA requires for the software and, if applicable, plan the release of the software through the Software Release Authority. This item should address how NASA will obtain and retain necessary intellectual property rights for the life of the software when the software includes components not written by civil servants (including but not limited to software written by contractors, off-the-shelf software, open source software, and auto-generated software).

q. **Training.** {A-C, S/C} Identify and plan project-specific training for software personnel.

r. **Operations and retirement.** {A-D} Describe the plan for software operations and retirement.
E.2 **Software Configuration Management Plan**

The Software Configuration Management Plan should include:

a. **Scope.** (A-D) Identifies the project and software items, to which the plan applies.

b. **Organization.** (A-D) The project organization(s) that participates in the software configuration management process. Describes roles, responsibilities and authorities for the software configuration management process.

c. **Configuration identification.** (A-D) Identify the software configuration items (e.g., software documents, code, and data) and their versions to be controlled for the project. Also include the scheme for the identification of software configuration items, establishment of baselines, and version references.

d. **Configuration control.** (A-D) Describe the process for submitting, evaluating, and approving changes to configuration items.

e. **Status accounting.** (A-C, S/C) Describe the process for preparing and maintaining records of the configuration status of configuration items.

f. **Configuration audits.** (A-C, S/C) Describe the process for auditing configuration items to determine the correct version of the configuration items and verify that the configuration items conform to the documents that define them.

g. **Release process.** (A-D) Describe the procedures for the storage, handling, delivery, release, and backup of deliverable software products and data.

h. **Schedule.** (A-D) Establish the sequence of configuration management activities and describe their coordination with the project's other plans.

i. **Resources.** (A-D) Identify the staffing, software tools, and equipment necessary to execute the configuration management plan.
E.3 Software Test Plan

The Software Test Plan should include:

a. **Scope.** {A-D} Identifies the project and software items, to which the plan applies.

b. **Organization.** {A-D} Identifies the project organizations that participate in the tests managed under this plan. Describes the roles, responsibilities, and authorities for developing and executing tests and for analyzing and reporting test results.

c. **Testing environment(s) and resources.** {A-C, S/C} Identify and describe the test environment(s) and personnel necessary to develop and execute tests and to analyze and report test results. Identify the software, hardware, and other materials that the project is responsible for obtaining or developing.

d. **Test Classification.** {A-C, S/C} Identify and describe the taxonomy of the planned tests. Tests may be classified by level, type, and class. A test level is a distinct test effort that has its own documentation and resources, e.g., system-level test versus software-level test. Test types are associated with the maturity of the software item and the scope of the test; examples include unit tests, software integration tests, systems integration tests, end-to-end tests, acceptance tests, and regression tests. A test class groups tests that exercise common aspects of the software (e.g., timing tests, erroneous input tests, maximum capacity tests, or safety-critical tests).

e. **Data recording, reduction, and analysis.** {A-D} Identify and describe the data recording, reduction, and analysis procedures to be used during and after the tests identified in this Software Test Plan.

f. **Planned tests.** {A-D} Identify the items to be tested and the test cases associated with each item. Summarize each test case to identify its purpose and to clearly differentiate the test cases. When a test taxonomy is described, each test case should be classified. If a project does not detail the test cases in a separate test procedure, then the project should summarize the test preparation, test input, test steps, and expected outcome in this Software Test Plan.

g. **Test schedules.** {A-D} Contain or reference the schedules for conducting the tests identified in this plan. Describe how test activities are coordinated with other project plans.

h. **Bidirectional requirements traceability.** {A-C, S/C} Trace each software requirement to the test case(s) that address it. Trace each test case to the software requirement(s) that it addresses.
E.4 Software Maintenance Plan

Maintenance is the process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment. [ISO/IEC/IEEE 24765:2017-09]

The Software Maintenance Plan defined here covers bug fixes and minor enhancements to a software release in operation. Development of a new major release (e.g., adds new features) should be conducted as new development, with software requirements, a budget, and a schedule, under a Software Management Plan. Furthermore, the plan described here is an extension of the Software Management Plan, the Software Configuration Management Plan, and, if applicable, Software Assurance Plan. Therefore, it does not repeat the content of those plans.

The Software Maintenance Plan should include the following:

a. **Scope.** (A-D) Identifies the project and software items, to which the plan applies.

b. **Organization.** (A-D) The project organization(s) that participate in the maintenance of the software. Describes roles, responsibilities and authorities for the software maintenance process.

c. **Maintenance process implementation.** (A-D) Describe the process for requesting maintenance. This item may include a) the process for user reporting of defects in operation and for submitting requests for minor enhancements, b) the process for notifying relevant stakeholders of requests for maintenance, and c) how status of maintenance requests are tracked.

d. **Problem and modification analysis.** (A-D) Describe how the project will analyze maintenance requests and disposition the requests. Analysis assesses the estimated cost and effort to implement a request, the impact of a request to software products, and the risks associated with a request.

e. **Modification implementation.** (A-D) Describes the process that transforms approved maintenance requests to verified implementations. In addition to modifying the software item, this process may include updates to the documented software requirements, software design documentation, software test documentation, and end-user documentation. The process should also describe how the software modification will be verified and validated to the same level of confidence as the original software.

f. **Maintenance acceptance.** (A-D) Describe the process for closing the maintenance request and transitioning the modified software to the operating environment.

g. **Migration.** (A-D) This item applies only when the project plans to migrate the software from an old operating environment to a new operating environment. Describe the process for modifying, verifying, and validating the software in the new environment to the same level of confidence as the old environment.
E.5 Software Requirements Specification

The Software Requirements Specification should include:

a. **Introduction.**\{A-D\} Identifies the software item whose requirements are defined in this specification. This item may also describe the relationship of this software requirements specification to other requirements specifications and list other documents referenced by this specification.

b. **System overview.**\{A-D\} Briefly state the purpose of the system and the software to which this document applies. This item may show the software in its operational context and summarize the concept of operations for the software. Consideration should be given to identify stakeholders in the software's requirements; stakeholders may include the project sponsor, acquirer, user (e.g., Principal Investigator and science team), developer, and maintenance organization.

c. **Requirements.**\{A-D\} Define the requirements for the software item(s). Requirements should be uniquely identifiable and accompanied by rationale. When developing requirements, the project should consider the following areas:
   - Required states and modes.
   - Functional requirements.
   - Performance and timing requirements.
   - External interface requirements.
   - Internal interface requirements.
   - Internal data requirements.
   - Testing requirements (e.g., peek-poke, checkpoint-restart).
   - Adaptation requirements (e.g., configurability for different installations or missions).
   - Safety requirements.
   - Security and privacy protection requirements.
   - Environment requirements (e.g., computer hardware and operating system).
   - Computer resource requirements.
   - Software quality characteristics.
   - Design and implementation constraints.
   - Personnel-related requirements (e.g., accommodations for user base).
   - Training-related requirements (e.g., built-in help or tutorials).
   - Logistics-related requirements (e.g., facilitate maintenance or deployment).
   - Packaging requirements (e.g., storage limits, cyclic redundancy checks, or digital signature).
   - Identify the precedence and criticality of requirements.

d. **Bidirectional traceability.**\{A-C, S/C\} NPR 7150.2 requires bidirectional traceability between software requirements and higher-level requirements, architecture, design, code, and test procedures.
E.6 **Software Architecture Description**

The software architecture description should include:

a. **Scope.** {A-C, S/C} Identifies the software item(s) defined in the document. Explains the purpose and scope of the Software Architecture Description, and indicates what information is and is not included. Also describe the relationship of this Software Architecture Description to other architecture descriptions.

b. **Conventions and notation.** {A-C, S/C} Identify the conventions and notations used to express the software architecture. For example, if the architecture is expressed using a viewpoint model such as DoDAF or a notation such as UML, that is described here.

c. **System overview.** {A-C, S/C} Summarizes the function and purpose of the software item(s). Depict the software in its operational context and summarize the concept of operations. Present the goals to be accomplished by the software architecture. Identify the relevant stakeholders and stakeholder concerns to be addressed by the architecture.

d. **Driving requirements.** {A-B, S/C} Describes the key functional and quality attribute requirements that shaped the software architecture.

e. **Architecture trades and analysis.** {A-B, S/C} Describe the architecture trades or analysis used to make architectural decisions.

f. **Architecture views.** {A-C, S/C} Present the software architecture using multiple views. A complete architecture typically provides three classes of views: structural or static (division of software into modules, components, units, classes, and/or objects, the relationships between these divisions, and the relationships between the software and the rest of the system), dynamic or behavioral (function call sequences, data flow, state charts; includes both internal interactions and external interactions), and allocated or deployment (runtime representation on hardware, allocation of compute resources to processes, threads, or partitions).

g. **Bidirectional traceability.** {A-C, S/C} NPR 7150.2 requires bidirectional traceability between the architecture and requirements and between the architecture and the detailed design.
E.7 Software Design Description

Note: In this section, a software item is decomposed into software units; software units may occur at different levels of a design hierarchy and may consist of other software units. A project may need multiple software design descriptions to cover all of the software items and units of the design.

Definitions:

Software item: An aggregation of software, such as a computer program or database that satisfies an end use function. [IEEE 24765:2010 3.2766]

Software unit: The lowest element in one or more software components. [IEEE 24765:2010 3.27809]

The Software Design Description describes the design of a software item and should include:

a. **Scope.** {A-C, S/C} Identifies the software item described in the document. Describe the relationship of this software design description to the software architecture description and to other software design descriptions.

b. **Conventions and notation.** {A-C, S/C} Identify the conventions and notations used to express the software design. For example, the design description could reference the Unified Modeling Language (UML) for its conventions and notations.

c. **Design decisions.** {A-B, S/C} Describe the design trades or analysis used to make design decisions. Describe rationale for design decisions.

d. **Architecture overview.** {A-C, S/C} Summarize the software architecture. Depict the software in its operational context including identification of all external interfaces and describe how the software item is divided into software units.

e. **Detailed design.** {A-C, S/C} Present the design of each software unit. The design description may address the following:
   - unit-level design decisions
   - design constraints
   - definition of inputs, outputs, and internal state including the initial state of the unit and response of the unit to inputs
   - definition of functions including signatures, preconditions, and post-conditions
   - the data or object model of the software unit
   - states and state transitions of the software unit
   - nominal sequence of functions
   - off-nominal design including detection and handling of errors, exceptions, or faults.

f. **Interface design.** {A-C, S/C} Describe the design of each external interface. The interface design description may address the following:
   - Identify the external element(s) on the other end of the interface.
   - Describe the roles of the software item and external element(s) with respect to the interface.
   - Define each datum exchanged over the interface (e.g. a sensor measurement).
   - Define the aggregation of data used exchanges over the interfaces (e.g., packets, records, messages, structures, files, arrays, reports).
   - Define the communication media or methods (e.g., Ethernet, 1553 Bus, S-Band Radio)
   - Define the exchange protocol(s) (e.g., Transmission Control Protocol/ Internet Protocol (TCP/IP), User Datagram Protocol (UDP)) including the description of any exchange logic (e.g. handshaking, encryption, and acknowledgement) layered on top of the protocol.
   - Interface compatibility if the software is required to support different revisions of the interface or support different options for instantiating the interface.

g. **Bidirectional traceability.** {A-C, S/C} NPR 7150.2 requires bidirectional traceability between the software design and the software requirements, architecture, and code.
E.8 **Software Test Procedures**

The Software Test Procedures should include:

a. **Scope.** {A-C, S/C} A software test procedure may detail the execution of multiple test cases. The scope identifies the test cases executed by the procedure and may describe the purpose for this selection of test cases.

b. **Test preparations.** {A-C, S/C} Describes the preparation of the test environment including hardware, software, and data.

c. **Test descriptions.** {A-C, S/C} For each test case or series of test cases, describe the test inputs, the instructions for executing the test, the method for analyzing results, and the expected outcome or other success criteria.

d. **Bidirectional traceability.** {A-C, S/C} NPR 7150.2 requires bidirectional trace between the software test procedures and the software requirements.
E.9 **Software Test Report**

Software Test Report should include:

a. **Software Item.** {A-C, S/C} Identifies the software item and its version that were tested.

b. **Summary of the test results.** {A-C, S/C} Summarizes the outcome of the tests covered in the report and, if the report documents problems, summarize the severity and impact of those problems.

c. **Detailed test results.** {A-C, S/C} For each test case, describe the results of the test and whether the test met the success criteria. Also, describe any deviations from the documented test procedure.

d. **Problems encountered.** {A-C, S/C} For each problem encountered, provide the identifier of the problem report used to track the problem, the identifier of the test case that uncovered the problem, and the step in the test procedure where the problem was encountered.

e. **Test log.** {A-C, S/C} For each test case, log the following information: the date and time the test was executed, the test location, the individual(s) who performed the test, and, if applicable, the names of test witnesses.
E.10 **Software Change Request**

The Software Change Request should include:

a. **Project.** {A-D} The project or software item, to which the change request is directed.
b. **Submitter.** {A-D} Name of the originator of Software Change Request.
c. **Submission date.** {A-D} Date change request submitted.
d. **Identifier.** {A-D} A unique identifier that can be used to reference the request.
e. **Change description.** {A-D} Description of change requested and rationale.
f. **Priority.** {A-D} The project can define priority using a rating scale (e.g., 1 through 5, low/normal/high), a requested completion date, assignment to a planned release, or other indication.
g. **Analysis of change.** {A-C, S/C} Analysis of the proposed change which may include a description of the proposed implementation, the configuration items (i.e., software units and documentation) that will require change, the impact of the proposed implementation to cost and schedule, impact of the change on project risks, and, for safety-critical software, any safety implications of the proposed change.
h. **Status.** {A-D} Status of change request which may include submitted, reviewed, duplicate, approved, disapproved, implemented, verified, and closed.
i. **Resolution.** {A-D} Description of the actions (including software modifications) taken to close the request.
E.11 Software Problem Report

The Software Problem Report should include:

a. **Software item.** (A-D) Identification of the software item exhibiting the problem. Identification may include the software item's version number.

b. **Submitter.** (A-D) Name of the originator of Software Problem Report.

c. **Report date.** (A-D) Date the problem was discovered and reported.

d. **Identifier.** (A-D) A unique identifier that may be used to reference the problem report.

e. **Problem description.** (A-D) A description of the problem. The description should capture necessary information to reproduce the problem including version identification of system components (hardware configuration, operating system, software item version, etc.) when the problem occurred. The description may also include suggested solutions or workarounds from the submitter.

f. **Severity.** (A-D) A rating scale, narrative, or other indicator of the impact of the problem to continued development or operations.

g. **Analysis of problem.** (A-D) Analysis of problem including an identification of the root cause, available workarounds, a description of the proposed solution, the configuration items (i.e., software units and documentation such as Software Design Description and Software Test Procedure) that will require modification, the impact of the proposed solution to cost and schedule, impact of the solution to project risks, and, for safety-critical software, any safety implications of the problem, proposed corrective action, or workaround.

h. **Status.** (A-D) Status of the problem report which may include submitted, reviewed, duplicate, approved, disapproved, implemented, verified, and closed.

i. **Resolution.** (A-D) A description of the actions (including software modifications or workarounds) taken to resolve the problem.
E.12 Software Version Description

The Software Version Description should include:

a. **Name of software configuration item and its version.** {A-D} Examples include Software X –Version 9.2, Software X-MM/DD/YY, Software X- Release 2. If the software deployed to multiple systems or is customized for a particular system, then a system identifier may also be included.

b. **Summary of changes.** {A-D} Summarize the updates or changes since the previous Software Version Description. This summary may include a list of change requests and problem reports newly resolved in this version.

c. **Known problems.** {A-C, S/C} List or summarize the problem reports that remain open. Provide any workarounds to known problems.

d. **Build and Installation instructions.** {A-D} Instructions for unpacking, building, installing, and/or verifying 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.

e. **Inventory.** {A-D} The list of software product files and their versions that compose the identified version of the software configuration item. Software product files may include machine code, source code or models, data, lifecycle documents (requirements, design, etc.), installation files (including batch files, command files, data files, other software needed to install the configuration item on its target system), and end-user documentation. For software configuration items whose official version copy resides in a configuration management system, this item may simply identify the mechanism (label, time stamp, workspace configuration, etc.) that can be used to obtain the inventory and will succeed in reproducing the identified version of the software configuration item.
E.13 **Software User Manual**

The Software User Manual should contain:

f. **Introduction.** {A-B, S/C} Identifies the software item and the software item version(s) to which the manual applies.

g. **Software summary.** {A-B, S/C} A brief description of the intended use of the software or its concept of operations; a summarized list of features.

h. **Warnings.** {A-B, S/C} Highlight warnings, cautions, and notes regarding actions by users (including system administrators) that could cause a failure, a hazard, or other undesirable outcome. Provide corrective actions and workarounds where applicable.

i. **Resource requirements.** {A-B, S/C} Identify the hardware, software, manual operations, and other resources needed for a user to install and operate the software.

j. **User access.** {A-B, S/C} Describes the procedure for first time users to access and launch the software. If the software includes access controls, describe the various roles, permissions, and user authentication features of the software.

k. **User procedures.** {A-B, S/C} Step-by-step instructions for performing tasks using the software. Shutdown procedures should be included as a task.

l. **Troubleshooting.** {A-B, S/C} Troubleshooting and error correction procedures.

m. **Problem reporting.** {A-B, S/C} Instructions for problem reporting and, if applicable, obtaining assistance (e.g. help desk services).

n. **Command reference.** {A-B, S/C} An organized list of all the software's user commands or the key-press, mouse-click, or touchscreen selections (e.g. keyboard shortcuts, menus, ribbons, dialogs, widgets). Provide a detailed explanation for each command.

o. **Message reference.** {A-B, S/C} An organized list of all the software's messages or return codes that are accessible to the user. Provide a detailed explanation of each message or return code.

p. **Version information.** {A-B, S/C} Information that is unique or specific to the version(s) of the software that this user manual covers. (E.g., new or modified features, new and modified interfaces).
Appendix F. LaRC Software Metrics Repository Data

Submit the measures at the milestones specified Table F.1 to the Langley Software Metrics Repository using the data entry spreadsheets provided at https://sw-eng.larc.nasa.gov/metrics-collection/. See the “Instructions” tab of the spreadsheet for instructions on delivering the data to the SEPG.

The measurement milestones are defined as follows:

- **Create Project**: Start of project formulation.
- **Start Development**: The Software Management Plan is approved.
- **End Development**: The software is completed and transitioned to operations and maintenance.
- **Start Maintenance**: The software is placed under maintenance.
- **Annual Maintenance**: The start of each fiscal year between Start Maintenance and End Maintenance.
- **End Maintenance**: The project is closed and the software is removed from service.

After software is released to operations, the Langley Software Metrics Repository treats each subsequent release of the software as a separate project and each subsequent release is entered into the Langley Software Metrics Repository beginning with the Create Project milestone. In the meantime, the software project continues to enter maintenance measures on the prior release until it is removed from service.

Software projects established solely to maintain heritage software start with the Create Project milestone and then move to the Start Maintenance milestone.

For each measure listed Table F.1, one of the following markers appears under each milestone:

- `<blank cell>`: The project does not enter the measure at this milestone.
- `X`: All projects enter the measure at this milestone.
- `A|B`: Projects developing Class A or B software enter the measure at this milestone.
- `#`: The project can change the value of the measure but is not required to enter a new value.

The “LaRC Measure Definitions” for each measure in the below table and the “Software Measurement Description for NASA Langley Research Center” that contains the rationale and use of each of the below measures are both at: https://sw-eng.larc.nasa.gov/supporting-products/.

<table>
<thead>
<tr>
<th>Measure</th>
<th>Entry Milestone for Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Name</td>
<td>X</td>
</tr>
<tr>
<td>Project Type</td>
<td>X</td>
</tr>
<tr>
<td>Application Domain</td>
<td>X</td>
</tr>
<tr>
<td>Software Class</td>
<td>X</td>
</tr>
<tr>
<td>Safety-Critical Determination</td>
<td>X</td>
</tr>
<tr>
<td>Primary Language</td>
<td>X</td>
</tr>
<tr>
<td>NASA Software Lead’s Name</td>
<td>X</td>
</tr>
<tr>
<td>NASA Software Lead’s Phone Number</td>
<td>X</td>
</tr>
<tr>
<td>NASA Software Lead’s E-mail Address</td>
<td>X</td>
</tr>
<tr>
<td>NASA Software Lead’s Org Code</td>
<td>X</td>
</tr>
<tr>
<td>CMMI Rated Development Organization</td>
<td>A</td>
</tr>
<tr>
<td>CMMI Rating</td>
<td>A</td>
</tr>
<tr>
<td>Actual Start Date</td>
<td>X</td>
</tr>
<tr>
<td>Planned End Date</td>
<td>X</td>
</tr>
<tr>
<td>Actual End Date</td>
<td>X</td>
</tr>
<tr>
<td>Total Planned Effort</td>
<td>X</td>
</tr>
</tbody>
</table>

Verify the correct revision before use by checking the LMS Web site.
<table>
<thead>
<tr>
<th>Measure</th>
<th>Create Project</th>
<th>Start Development</th>
<th>End Development</th>
<th>Start Maintenance</th>
<th>Annual Maintenance</th>
<th>End Maintenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Planned Cost</td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Actual Effort</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Actual Cost</td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Planned Effort for Fiscal Year</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Planned Cost for Fiscal Year</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actual Effort for Fiscal Year</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actual Cost for Fiscal Year</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Requirements</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Requirements Verified</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Defects Reported</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Defects Resolved</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Defects Open</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actual Software Size(^5)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Most Important Constraint</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LMS Improvement Suggestion</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project's Greatest Challenge</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Best Practices</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(^5\) The following provides a link to the University of Southern California public code counters: [http://sunset.usc.edu/research/CODECOUNT/](http://sunset.usc.edu/research/CODECOUNT/)

Verify the correct revision before use by checking the LMS Web site.
Appendix G. Simplified NPR 7150.2B for Class D and E (Not Safety-Critical)
For the convenience of projects developing software that is Class D or Class E and not safety-critical, the remaining NPR 7150.2B requirements that flow down to the development team are extracted and presented here in an information mapping format with LaRC-specific guidance. This format presents a notional process flow followed by a step-action table defining each step of the flow. Separate step-action tables are presented for Class D and Class E since Class D must address a larger subset of NPR 7150.2B requirements. LaRC projects may use this Appendix as a substitute for NPR 7150.2B.
Verify the correct revision before use by checking the LMS Web site.
## Step-Action Table for Class E

<table>
<thead>
<tr>
<th>STEP</th>
<th>ACTION TO TAKE</th>
</tr>
</thead>
</table>
| 1    | **LPR 7150.2B Project Formulation**  
Chapter 2 and Chapter 3 are performed before or concurrent with the next two steps. These chapters cover software classification and safety-critical determination; creation and approval of the NPR 7150.2B Compliance Matrix including any tailoring of procedural requirements; defining software engineering documents and records for the project; the make vs. buy decision; and acquisition. These topics cover inherently governmental activities; the remaining steps in this procedure can be performed by NASA or its suppliers. [SWE-125][SWE-145][SWE-020][SWE-132][SWE-133][SWE-021][SWE-160][SWE-033][SWE-035] |
| 2    | **Develop Requirements**  
The Project Software Manager shall record and maintain a description of the software needs with sufficient detail to inform implementation. [SWE-050]  
Guidance: Class E projects have freedom to informally develop and maintain a description of the software's expected form and behavior. This description may take the form of goals, use cases, concepts of operations, features, user stories, tasks, or requirements. |
| 3    | **Plan the Project**  
a. The Project Software Manager shall develop and maintain a software management plan (SMP) – OR – the Project Software Manager may place the project under a project portfolio and update the portfolio's SMP as required. [SWE-013]  
Guidance: See section 2.7 for more information on project portfolios. If the project will be managed under an individual software management plan, then the designated Engineering Technical Authority defines the contents of the plan per section 2.5. Recommended content for a software management plan appears in appendix E.1.  
b. The Project Software Manager’s Supervisor shall review and approve the SMP to ensure the project is in compliance with LPR 7150.2B and to approve the allocation of the Supervisor’s staff and resources (e.g., licenses, travel, etc.). |
| 4    | **Execute Plans and Track Software Activities**  
a. The Project Software Manager shall execute the SMP. [SWE-013]  
b. The Project Software Manager shall replan the project as warranted by changes to the project’s scope, budget, or schedule. [SWE-013]  
Guidance: Per section 2.2, the NASA Software Lead will determine if project changes elevate the software classification or cause the software to become safety-critical. Should this occur, the NASA Software Lead will direct the Project Software Manager to follow procedural requirements for the new class or safety-criticality. |
| 5    | **Engineer Software**  
a. The Project Team develops software that fulfills the requirements.  
Guidance: The Project Team is encouraged to use version control so that the software can be returned to a working state when new changes break functionality. The Project Team is also encouraged to document design and implementation decisions in code/model comments or in a journal (e.g., log, blog, wiki) in case work on the software is suspended or reassigned.  
b. The Project Team shall perform software testing. [SWE-066]  
Guidance: Use software testing to prove that the software fulfills the requirements. |
| 6    | **Complete Software**  
Deliver the completed software product(s) and/or data to the customer with appropriate documentation. [SWE-077]  
Guidance:  
1. The appropriate documentation is negotiated between the customer and the project.  
2. To reconstruct the software product(s) and/or data at a future date, the Project Team should document how to build and execute the software product(s). |
## G.2 Step Action Table for Class D

<table>
<thead>
<tr>
<th>STEP</th>
<th>ACTION TO TAKE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>LPR 7150.2B Project Formulation</strong></td>
</tr>
<tr>
<td></td>
<td>Chapter 2 and Chapter 3 are performed before or concurrent with the next two steps. These chapters cover software classification and safety-critical determination; creation and approval of the NPR 7150.2B Compliance Matrix including any tailoring of procedural requirements; defining software engineering documents and records for the project; the make vs. buy decision; and acquisition. These topics cover inherently governmental activities; the remaining steps in this procedure can be performed by NASA or its suppliers. [SWE-125][SWE-145][SWE-020][SWE-132][SWE-133][SWE-021][SWE-033][SWE-022][SWE-036][SWE-037][SWE-039][SWE-046]</td>
</tr>
<tr>
<td>2</td>
<td><strong>Develop Requirements</strong></td>
</tr>
<tr>
<td></td>
<td>The Project Software Manager shall establish, capture, record, approve, and maintain software requirements, including the software quality requirements. [SWE-050]</td>
</tr>
<tr>
<td></td>
<td>Guidance: The designated Engineering Technical Authority defines the contents of requirements documentation per section 2.5. Center recommended content for a software requirements specification appears in appendix E.5.</td>
</tr>
<tr>
<td>3</td>
<td><strong>Plan the Project</strong></td>
</tr>
<tr>
<td></td>
<td>a. The Project Software Manager shall select and document a software development life cycle or model that includes transition criteria for each life cycle phase or each iteration or increment of development. [SWE-019]</td>
</tr>
<tr>
<td></td>
<td>Guidance:</td>
</tr>
<tr>
<td></td>
<td>1. Development life-cycles or models define type and order of software engineering activities to be performed. As such, they influence the cost, schedule, and plans that the Project Software Manager develops in the steps that follow.</td>
</tr>
<tr>
<td></td>
<td>2. Traditional lifecycle phases include the requirements phase, design phase, coding phase, integration phase, test phase, acceptance phase, and maintenance phase.</td>
</tr>
<tr>
<td></td>
<td>3. The development life cycle or model is often documented in the software management plan, see step 3.d.</td>
</tr>
<tr>
<td></td>
<td>b. The Project Software Manager shall establish, document, and maintain a software cost estimate that satisfies the following criteria: [SWE-015][SWE-151]</td>
</tr>
<tr>
<td></td>
<td>1. Covers the entire software life cycle.</td>
</tr>
<tr>
<td></td>
<td>2. Is based on selected project attributes (e.g. estimated software size).</td>
</tr>
<tr>
<td></td>
<td>3. Is based on the cost implications of the technology to be used and the required maturation of that technology (i.e., the cost of increasing the Technology Readiness Level [TRL] of new technologies).</td>
</tr>
<tr>
<td></td>
<td>4. Includes other direct costs (e.g., travel, training, off-the-shelf software or hardware purchases, or fixed-fee services).</td>
</tr>
<tr>
<td></td>
<td>Guidance: The software cost estimate is typically documented in a software management plan, see step 3.d. Except for other direct costs, the Project Software Manager can establish the software cost estimate using labor hours instead of dollars.</td>
</tr>
<tr>
<td></td>
<td>c. The Project Software Manager shall document and maintain a software schedule that satisfies the following criteria: [SWE-016]</td>
</tr>
<tr>
<td></td>
<td>1. Coordinates with the overall project schedule (if the software is an element of a larger project).</td>
</tr>
<tr>
<td></td>
<td>2. Documents the interactions of milestones and deliverables between software, hardware, operations, and the rest of the system.</td>
</tr>
<tr>
<td></td>
<td>3. Reflects the critical path for the software development activities.</td>
</tr>
<tr>
<td></td>
<td>d. The Project Software Manager shall develop software plans that cover the entire software lifecycle:</td>
</tr>
<tr>
<td></td>
<td>1. A software management plan (SMP) [SWE-013]</td>
</tr>
<tr>
<td></td>
<td>2. A software test plan (STP) [SWE-065]</td>
</tr>
<tr>
<td></td>
<td>3. A software maintenance plan, if the software will be maintained after first use [SWE-075]</td>
</tr>
<tr>
<td></td>
<td>4. Other software plans as determined by the designated Engineering Technical Authority</td>
</tr>
</tbody>
</table>

**Verify the correct revision before use by checking the LMS Web site.**
<table>
<thead>
<tr>
<th>STEP</th>
<th>ACTION TO TAKE</th>
</tr>
</thead>
</table>
| (per section 2.5.1). [SWE-036][SWE-153] | Guidance:  
1. See Appendix E for guidance on plan documentation.  
2. All plans do not need to be developed at project start. For example, the software test plan can be developed concurrently to software implementation.  
3. The designated Engineering Technical Authority defines the type and content of software plans required for the project per section 2.5.  
4. The NASA Software Lead is responsible for inserting requirements for software plans into solicitations and final agreements when software activities are delegated to suppliers, see section 2.6.5.  
5. Multiple class D projects within the same domain can be jointly managed as a project portfolio. The Project Software Manager may simply place the project in the portfolio and update the portfolio's software plans as required. See section 2.7 for more information. |
| e. Where documented and approved in the NPR 7150.2B Compliance Matrix, the Project Software Manager shall document and reflect the tailoring of procedural requirements in the plans or procedures controlling the development and/or deployment of the software. [SWE-121] | Guidance: Development of the NPR 7150.2B Compliance Matrix and tailoring of procedural requirements is addressed in sections 2.3, 2.4, and 3 and in appendix D. Procedural requirements within this table (i.e., those with a [SWE-XXX] reference) may be among those tailored in the NPR 7150.2B compliance matrix. The tailoring takes precedence. |
| f. The Project Software Manager shall address the following topics in the software management plan: |  
1. How the Project Software Manager will track the actual results and performance of software activities, including cost and schedule, against the software plans [SWE-024]  
Guidance: Some development methods use proxies for tracking cost or schedule. For example, Scrum teams may use story points as a proxy for cost and velocity as proxy for schedule.  
2. What regular meetings the Project Software Manager will hold with stakeholders to review software activities, status, and results [SWE-018]  
Guidance: Stakeholders are individuals that are affected by or in some way accountable for the outcome of the project (may include project members, suppliers, customer / acquirer, end users, and others). Status meetings and major milestone reviews typically serve this purpose.  
3. How issues raised in stakeholder meetings are captured and tracked to resolution [SWE-018]  
4. How requirements changes will be tracked and managed [SWE-053]  
Guidance: The project could document requirements changes by: a) keeping a revision history or log of the changes to the software requirements specification, or b) retaining revision marked versions of the software requirements specification, c) implementing a change request system, d) utilizing a product backlog (e.g., Scrum), or e) using a requirements management tool like DOORS. The Project Software Manager should make all stakeholders aware of requirements changes.  
5. How software changes will be evaluated and tracked [SWE-080]  
Guidance: This is traditionally done with a change request system or a problem tracking system.  
6. Identify the software configuration items (e.g., software records, code, data, tools, models, scripts) and their versions to be controlled for the project. [SWE-081]  
7. Procedures for the storage, handling, delivery, release, and maintenance of deliverable software products [SWE-085]  
Guidance: Large projects may need to develop a software configuration management plan to address items 4 through 6.  
8. How software defects will be recorded and tracked to closure [SWE-089]  
Guidance: Defects may also be called “bugs”, “problem reports”, or “corrective actions”. Appendix E.11 provided recommended content for Software Problem Reports. An electronic tracking system is commonly used to record defects and track their progress to |
<table>
<thead>
<tr>
<th><strong>STEP</strong></th>
<th><strong>ACTION TO TAKE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>closure. However, small projects can also use &quot;to-do lists&quot; (e.g., in Microsoft Outlook), sticky notes or index cards on a wall or other dedicated surface, or product backlogs (when using an Agile method like Scrum).</td>
</tr>
</tbody>
</table>
| g. | The Project Software Manager shall address the following topics in the software test plan:  
1. **Software verification activities, methods, environments, and criteria for the project.** [SWE-028] Guidance: Verification confirms that the software satisfies the requirements. A requirement can be verified by test, inspection, analysis, or demonstration.  
2. **Software validation activities, methods, environments, and criteria for the project.** [SWE-029] Guidance: Validation confirms that the software fulfills its intended use. Validation evaluates the software using scenarios derived from use cases, concepts of operations, stakeholder expectations, etc. As with verification, validation methods can also be categorized as test, inspection, analysis, or demonstration. Examples of validation methods are scenario walkthroughs, peer reviews, prototype or product demonstrations, deployment to representative users in an alpha or beta test, and running the software through scenarios in the target environment or a facsimile (e.g., simulation).  
3. **How the project will perform software unit testing.** [SWE-062] Guidance: Unit testing evaluates a software unit against the requirements allocated to that unit. Unit testing may also assess the software unit using quality criteria such as subjecting the software unit to the full range of inputs, including both valid and invalid inputs, and analyzing the unit's response. To be effective, unit testing requires either a software design that permits isolated testing of software units or requires that sections of the code or model are tested in isolation before the code or model is added to the software product. Not all software units are suitable for unit testing since some fraction of units, by necessity, must coordinate the interactions among other units and, therefore, cannot easily be isolated. Unit testing detects defects early, while the team is coding or modeling. However, unit testing is less effective at confirming that the finished product will meet requirements (verification) or fulfill its intended use (validation).  
4. **The software test cases to be used for verification or validation.** [SWE-065] Guidance: A "test case" is a less formal substitute for "test procedures" required in NPR 7150.2 SWE-065. See Appendix E.3.f for recommended information to provide for each test case.  
5. **How the results of verification and validation activities will be recorded.** [SWE-030][SWE-031][SWE-065][SWE-068] Guidance: A minimalist approach for Class D projects is to simply record a pass-fail status for V&V activities. For example, this can be done in a test matrix implemented as a spreadsheet, where each row lists a test case and one column contains pass-fail status. |
| h. | The software maintenance plan shall address software operations, maintenance, and retirement activities [SWE-075] |
| i. | The Project Software Manager’s Supervisor shall review and approve the SMP to ensure the project is in compliance with LPR 7150.2B and this procedure and to approve the allocation of the Supervisor’s staff and resources (e.g., licenses, travel, etc.). Guidance: Other plans do not require supervisory approval, but may also need the approval of select stakeholders. |
| 4 | **Execute Plans and Track Software Activities**  
a. The Project Software Manager shall ensure that the Project Team operates in accordance with the software plans. [SWE-013]  
b. The Project Software Manager shall track the actual results and performance of software activities against the software plans. [SWE-024]  
c. The Project Software Manager shall regularly hold reviews of software activities, status, and results with the project stakeholders and track issues to resolution. [SWE-018]  
d. The Project Software Manager shall track and manage changes to the software requirements. [SWE-053] |

Verify the correct revision before use by checking the LMS Web site.
<table>
<thead>
<tr>
<th>STEP</th>
<th>ACTION TO TAKE</th>
</tr>
</thead>
<tbody>
<tr>
<td>e.</td>
<td>The Project Software Manager shall update the software test plan to be consistent with the software requirements. [SWE-071]</td>
</tr>
<tr>
<td>f.</td>
<td>The Project Software Manager shall track and evaluate changes to software products. [SWE-080]</td>
</tr>
</tbody>
</table>
| g.   | The Project Software Manager shall cooperate with spot checks of the software development process conducted by the Mission Assurance Branch (MAB). [SWE-022]  
Guidance: NASA-STD-8739.8 NASA Software Assurance Standard allows substantially tailoring of its requirements for Class D software. LaRC Engineering and LaRC Mission and Safety Assurance have agreed that the minimal software assurance activities for Class D projects includes spot checks of the software development process by MAB. The other software assurance activities for Class D projects are covered by Chapter 2. |

## 5 Engineer the Software

a. The Project Team develops software that fulfills the requirements. [SWE-060]  
Guidance: Class D, not safety-critical projects are free to determine how they will realize the software requirements into an executable software product. However, the Center recommends that, when software will be put to continuous use such as new engineering tools, decision support tools, or operational software for labs, the project team develop and document the software design during software development. |

b. The Project Team shall perform unit testing per the software test plan. [SWE-062] |
c. The Project Team shall perform verification and validation, to include software testing, as defined in the software test plan. [SWE-028][SWE-029][SWE-066]  
Guidance: In this step, the project team verifies and validates the integrated software product. |
d. The Project Team shall evaluate and record the results of verification and validation activities as defined in the software test plan. [SWE-030][SWE-031][SWE-065][SWE-068] |
e. The Project Team shall record defects identified during verification and validation activities and track the defects to closure. [SWE-030][SWE-031][SWE-069] |
f. The Project Team shall evaluate the software against its acceptance criteria and conditions. [SWE-034]  
Guidance: This may not be distinct from verification and validation; it is common for acceptance criteria to be incorporated into verification and validation (V&V) activities. However, acceptance criteria can also include items not addressed by V&V such as completion of deliverable documentation (e.g. user’s manual). Acceptance criteria is defined per section 2.5.8. |

## 6 Complete the Software

a. Provide a software version description for each software delivery to the requester or intended user. [SWE-063]  
Guidance: A software version description will be produced whenever software is created or modified for delivery or to produce deliverable data. The designated Engineering Technical Authority defines the contents of the software version description per section 2.5. Center recommended content for the software version description appears in appendix E.12. |
b. Complete and deliver the software product to the customer with appropriate documentation to support the operations and maintenance phase of the software's life-cycle. [SWE-077]  
1. However, if the code is not a deliverable (i.e., only data or analysis are delivered or published), the project documents and retains the instructions to build the software, execute the software, and recreate the data or analysis. [SWE-077]  
Guidance: This documentation may be included in the software version description. |