CLASS G Software Requirements (General Purpose Computing, Business and IT Software (Single Center or Project))

1. The Contractor shall develop, execute, and maintain a Software Management Plan (SMP) as defined in Appendix A1 (of this document).

2. The Contractor shall develop and maintain the following supporting documents as defined in the following appendices (of this document):

    a. Software Configuration Management Plan as defined in Appendix A2
    b. Software Requirements Specification as defined in Appendix A3
    c. Software Design Description as defined in Appendix A4
    d. Software Test Plan(s) as defined in Appendix A5
    e. Software Version Description, as defined in Appendix A6, for each software release

    Guidance: The documents above may be combined or kept separate. Required content may be included in project level documents.

3. The Contractor shall submit the supporting documents to the Contracting Officer for review and approval in accordance with the schedule in the approved SMP.

4. The Contractor shall execute approved plans and procedures.

5. The Contractor shall perform peer reviews on software products per the approved Software Management Plan.

6. The Contractor shall participate in any identified joint NASA/contractor audits of the software development process and software configuration management process.

7. The Contractor shall provide all software deliverables in electronic format.

8. The Contractor shall provide electronic access to the source code developed/modified for the project, including modified off-the-shelf software.

9. The Contractor shall flow down all software contractual requirements to all levels of software supplier sub-contracts.

10. The Contractor shall notify NASA, whether off-the-shelf, reused, or open-source software will be included in code developed for the project.

11. The Contractor shall satisfy the following conditions when a COTS, GOTS, MOTS, or reused software component is acquired or used:
    a. The requirements to be met by the software component are identified.
    b. The software component includes documentation to fulfill its intended purpose (e.g., usage instructions).
    c. Proprietary rights, usage rights, ownership, warranty, licensing rights, and transfer rights have been addressed.
    d. Future support for the software product is planned and adequate for project needs.
    e. The software component is verified and validated to the same level required to accept a similar developed software component for its intended use.

f. The project has a plan to perform periodic assessments of vendor reported defects to ensure the defects do not impact the selected software components.

12. **The Contractor shall provide insight into software development and test activities. The Contracting Officer will be allowed to:
    a. Monitor product integration.
    b. Review the verification activities to ensure adequacy.
    c. Review trades studies and source data.
    d. Audit the software development process.
    e. Participate in software reviews and systems and software technical interchange meetings.

13. ** The Contractor shall track software changes and non-conformances and provide the data for the project's review.

14. **The Contractor shall establish, capture, record, obtain approval of, and maintain software requirements, including the software quality requirements.

15. **The Contractor shall track and manage changes to the software requirements.

16. **The Contractor shall perform requirements validation to ensure that the software will perform as intended in the customer environment.

17. *The Contractor shall develop, record, and maintain the software design.

18. The Contractor shall verify the implementation of each software requirement.

19. The Contractor shall perform software testing, including unit testing.

20. *The Contractor shall record defects identified during testing and track to closure.

21. *The Contractor shall evaluate test results and record the evaluation.

22. The Contractor shall validate the software system on the targeted platform or high-fidelity simulation.

23. The Contractor shall deliver the completed software product(s) and/or data to the customer with appropriate documentation to support the operations and maintenance.

24. The Contractor shall establish and implement procedures for the storage, handling, delivery, release, and maintenance of deliverable software products.

25. The Contractor shall provide access to the software measurement data, measurement analyses, and software development status as requested by the Contracting Officer.

*For new software activities over $50K
**Applicable for projects over $50K

Appendix A1: Software Management Plan (SMP)

Instructions: The SMP is a living document. Therefore, some of the plan's required content may not be known at the time of its initial release and approval. However, for unknown items, expected closure dates should be assigned and tracked. Content may be rolled out to other documents (e.g., schedule).

The Software Management Plan shall contain: [SWE-102] [SWE-013]

a. Project Title and Description.
b. Project Organization. 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). [SWE-102.a] [SWE-103.a]
c. Software Classification and Safety Criticality. The software classification of each of the systems and subsystems containing software and the safety-criticality determination. (e.g., All the systems containing software on the project are Class G and not safety-critical.) [SWE-102.b] [SWE-020]
d. Engineering environment. [SWE-102.d] Describe software, hardware, and facilities necessary to develop the software products and, as applicable, those items necessary to operate and maintain the software products. This may include computing platform, operating system, libraries (Linear Algebra Package (LAPACK)), equipment (logic analyzer, emulators), standards (Unified Modeling Language Version 2.3, ISO/IEC 14882: Programming Language-C++), procedures (use of software development folders to maintain current development progress), and tools (complier, debuggers, static analyzers).
e. WBS, Schedule, Effort, and Cost. [SWE-102.e] [SWE-016] [SWE-015]
    1. Document a list (e.g., Work Breakdown Structure (WBS)) of: [SWE-102.e]
        (a) Software activities (e.g., developing plans, defining the requirements, developing the design, developing the code, implementing the tests, deliver of products/data, perform acceptance activities). [SWE-036] [SWE-102.e]
        (b) Software products that will be produced (e.g., Software Management Plan, Software Configuration Management Plan, Software Requirements Specification, Software Design Description, Software Test Plan, Software Version Description, software code, software data or analysis results, and Software Users Manuals). [SWE-102.e] [SWE-077]
        (c) Appropriate software documentation to be produced to support the operations and maintenance phases of the life cycle (e.g., Software Version Description that includes the software build and execution instructions, and the up-to-date versions of the appropriate software products listed above in e.1.b). [SWE-077] [SWE-102.e] The project should also consider developing a Software User Manual for the completed software; The extent of documentation "to support the operations and maintenance" will depend on the items delivered. For example, a delivered executable may require little or no "build" documentation. Documentation can take the form of text, audiovisual instructions, interactive scripts, help files, man pages, embedded instructions (e.g., invoked by passing a "help" argument to the software), etc.
        (d) [optional] Software services and non-deliverable items to be performed. [SWE-102.e]
    2. Document the software schedule associated with the activities, services, and deliverables that satisfies the following minimum conditions: [SWE-016] Note: the "Wide-Ban Delphi Estimation Process" at: https://sites-e.larc.nasa.gov/sweng/supporting-products/ provides an example process for estimating size, cost, or other types of estimates.
        (a) Coordinates with the overall project schedule if the software is an element of a larger project [SWE-016.a]
        (b) Documents the schedule for the activities. [SWE-102.e]
        (c) Documents milestones and delivery dates. [SWE-016.b] [SWE-102.e]
        (d) [optional] As part of acquisitions activities, documents milestones at which the software supplier(s) progress will be reviewed by the project or reference the contractual agreement where the milestones are defined. [SWE-037]
    3. Document an estimate of effort (include both civil servant and contractor effort) that covers either: [SWE-015]

(a) The entire software life cycle, or [SWE-015.a]
(b) If the software activity is ongoing, then the effort estimate may be for a period of performance (e.g., one year) rather than the whole life cycle. [SWE-015.a]

4. [optional] Document an estimate of special costs that covers the entire software life cycle or period of performance. [SWE-015.a] Special costs do not include labor and can include (but are not limited to) travel, training, and OTS software or hardware purchases and installation fees. It does not include items provided to but not purchased by the project, which may include developer workstations and software development tools.

f. Requirements Management. Describe how the project will collect and manage changes to the software requirements. [SWE-053][SWE-102.g] The project could document requirements changes either by: a) keeping a revision history of the changes to the Software Requirements Specification (e.g., Microsoft Word revision marks), or b) keeping old hand-marked versions, or c) implementing a change request system. Managing changes includes making all stakeholders aware of the changes.

g. Verification and validation plan. [SWE-102.i] [SWE-028] [SWE-029]

1. Document a verification and validation plan for the following: [SWE-028] [SWE-029]
   (a) Activities, methods (test plans and procedures, Software Inspections, analysis, demonstrations). [SWE-028] [SWE-029]
      - Examples of verification methods include, but are not limited to, Software Inspections, use of simulations, black box and white box testing techniques, software load testing, software stress testing, software performance testing, decision table-based testing, functional decomposition-based testing, acceptance testing, path coverage testing, analyses of requirement implementation, and software product demonstrations. [NPR 7150.2A:2.4.1 modified] Note that the project specifies the "verification methods" for each requirement in the Software Requirements Specification.
      - Examples of validation methods include, but are not limited to, prototype demonstrations, functional demonstrations, software testing, Software Inspections, behavior in a simulated environment, acceptance testing against mathematical models, analyses, and operational environment demonstrations. [NPR 7150.2A:2.4.2 modified]
   (b) Verification and validation criteria. [SWE-028] [SWE-029] [*Refer to Software Test Plan, where some or all of the criteria may be documented*].
   (c) Verification and validation environments (software/hardware platform) if different than the environment defined above in "Engineering environment." [SWE-028] [SWE-029] [SWE-034]

h. Software acceptance.

1. Document the acceptance criteria that will be used to confirm that the software fulfills its intended use (and any necessary conditions under which the acceptance criteria must be met). [SWE-029][SWE-034][SWE-102.i] The acceptance criteria and conditions may be documented in this plan, the Software Test Plan and/or in the contractual agreement. Examples of necessary conditions are operating under maximum loads such as maximum number of simultaneous users or peak bandwidth capacity; and operational modes such as startup, operations, shutdown, and maintenance. The following are suggested examples of acceptance criteria and conditions:
   (a) Complete a review of the Software Test Plan to ensure:
      (1) It is up to date with the latest version of the Software Requirements Specification,
      (2) That the plan contains the necessary and sufficient tests to make sure that the product fulfills all requirements in the Software Requirements Specification (and, if applicable, all requirements in the contractual agreement), and
      (3) That the plan covers tests and analysis to make sure that the software's actual output meets the expected output for execution under 1) minimum load/usage, 2) normal load/usage, and 3) maximum load/usage.
   (b) Make sure that a review of the final versions of all the documents has been performed to ensure that they are complete, spell checked, up to date, and accurate.
   (c) The source code compiles on the target with no syntax and no logical errors.
   (d) The software passes all tests in the Software Test Plan on the target computer.
   (e) The users report zero defects in X days of use.

2. Describe how the project will ensure software products meet acceptance criteria and any conditions (e.g., by performing acceptance testing, final customer checkout or readiness review).[SWE-029] [SWE-102i] For example:
   (a) Successfully compile the software on the target computer with no syntax and no logical errors.
   (b) As part of acceptance testing, execute the Software Test Plan on the target computer.
   (c) The customer performs "hands-on" testing by exercising the software as users for a period of X-days prior to final acceptance.
3. Describe how the project will record results of software acceptance, and address, and track issues to closure (e.g., a test log, defect tracking system, or validation matrix). [SWE-031] [SWE-102.i] [SWE-069] For example:
   (a) All identified defects will be documented in the test matrix.
   (b) Personnel will be assigned to:
      (1) Correct each defect and test each correction,
      (2) Update the Software Test Plan to include any new tests that were created to test the corrections, and
      (3) When complete, record the completion in the test matrix along with the version of the software and if changed, the version of the Software Test Plan.
   (c) The customer will witness the final testing.
   (d) Final testing will consist of re-executing the Software Test Plan on the target computer to validate that all identified errors have been shown to be corrected and that no new errors have been introduced by the corrections.
4. [optional] If applicable, describe how the project will perform any additional activities (other than testing) such as demonstration, analysis, or inspection to confirm the product fulfills its intended use and requirements; and how the project will record results of those activities, and address, and track issues to closure. [SWE-102.i]
   [SWE-028] [SWE-029] [SWE-031] For model and simulation software, information regarding specific verification and validation techniques and the analysis of models and simulations can be found in NASA-STD-7009, Standard for Models and Simulations (e.g., numerical accuracy, uncertainty analysis, and sensitivity analysis, as well as verification and validation for software implementations of models and simulations).

i. Reviews. [SWE-018]
   1. Define the reviews that the project will hold regularly of software activities, status, and results with project stakeholders and the frequency or schedule for the reviews.[SWE-018] [SWE-102.i] 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).* [Based on CMMI-DEV, V1.3]
   2. Describe how the project will track issues from reviews to resolution. [SWE-018] [SWE-102.i]

j. Plan Tracking. Describe how the project will ensure that actual milestones and deliverables are tracked against the SMP. [SWE-024] [SWE-102.o] This may be done as part of performing item i above.

k. Software development life-cycle phases. [SWE-019] [SWE-102.q] The phases can be included as part of the WBS. The following are typical life-cycle phases: requirements phase, design phase, coding phase, integration phase, test phase, acceptance phase, and maintenance phase. The project may also want to document the software life-cycle model to be used on the project (e.g., waterfall, spiral, agile). For additional guidance on choosing a life cycle, refer to "Guidance on Selecting Life Cycle and Development Approach" at: https://sites-e.larc.nasa.gov/sweng/supporting-products/ .

l. Configuration Management. Include here or provide the reference to the Software Configuration Management Plan. [SWE-102.r]

m. Maintenance, Describe how the project will maintain software products and data. [SWE-075] [SWE-085]

n. Off-the-shelf (OTS) software. Management, development, and testing approach for handling any commercial-off-the-shelf (COTS), government-off-the-shelf (GOTS), modified-off-the-shelf (MOTS), reused, or open-source software component(s) that are included within a NASA system or subsystem. [SWE-102.x] Also, address any proprietary, usage, ownership, warranty, licensing rights, transfer rights and any future support for the software product. [SWE-027.c] [SWE-027.d]

Appendix A2: Software Configuration Management Plan

Instructions: Software Configuration Management Plan may be included in the Software Management Plan or rolled out as a separate document.

The Software Configuration Management Plan shall contain: [SWE-013] [SWE-079] [SWE-103]

a. The project name. [SWE-103.a]
b. Assigned responsibilities and authority for the implementation of software configuration management on the project. [SWE-103.b] [SWE-079]
c. References to the software configuration management policies and directives that apply to the project. [SWE-103.c]
d. Describe the functions and tasks required to manage the configuration of the software, including the following: [SWE-079] [SWE-103.d]
    1. Describe how the project will identify and record the software configuration items (e.g., software documents, code, data, tools, models, scripts) and their versions to be controlled for the project, and [SWE-081] [SWE-079] [SWE-103.d]
    2. Describe how the project will track changes to software products. [SWE-079] [SWE-080] [SWE-103.d]
e. Schedule information, which establishes the sequence and coordination for the identified activities. [SWE-103.e]
f. Identification of any configuration management tools used. [SWE-103.f]
g. Plan maintenance information, which identifies the activities and responsibilities necessary to keep the Software Configuration Management Plan up to date. [SWE-103.g] This could be something as simple as stating: “On an as-needed basis, changes to the Software Configuration Management Plan will be made and submitted to the Software Manager for approval.”
h. Define how the project will store, back up, and deliver/release deliverable software products and data. [SWE-085] [SWE-103.h]

Appendix A3: Software Requirements Specification

Instructions: The following defines the required content, but the format and style are left to the project.

The Software Requirements Specification shall contain: [SWE-049] [SWE-109]

a. System overview. [SWE-109.a] Briefly state the purpose of the system and the software to which this document applies. Identify the project sponsor, acquirer, user (e.g., Principal Investigator and science team), developer/maintainer, and other stakeholders; identify current and planned operating sites; and list relevant documents. [J-STD-016:F.2.4(1)]
b. Functional requirements and data requirements (e.g., this includes the functions the software must perform and the required inputs, processing, and outputs). [SWE-109.b.1] A "function" is a group of related requirements. Include those characteristics of the software item that are conditions for software item acceptance, and defer to the Software Design Descriptions those characteristics that the customer is willing to leave up to the developer. Section d. of this Software Requirements Specification provides a list of topics to be considered when specifying requirements regarding input the software item is required to accept and output it is required to produce. [J-STD-016:F.2.4(3.2)]
c. [optional] Required states and modes. [SWE-109.b.2] Examples of states and modes include: startup, calibration, housekeeping, idle, ready, active, post-use analysis, training, degraded, backup. The distinction between states and modes is arbitrary. A software item may be described in terms of states only, modes only, states within modes, modes within states, or any other scheme that is useful. If states and/or modes are required, each requirement or group of requirements in this specification should be correlated to the states and modes. The correlation may be indicated by a table or by annotation of the requirements. [J-STD-016:F.2.4(3.1)]
d. [optional] External interface requirements. [SWE-109b.3]
    1. Identify the interfacing entities (systems, hardware items, software items, users, etc.) and interfaces to which this document applies. One or more interface diagrams may be included to graphically depict the interfaces. [J-STD-016:F.2.3(3.1)] Conventions needed to understand the diagrams should be presented or referenced. If the interface operates differently in different states or modes, each requirement or group of requirements for the interface should be correlated to the states and modes. [J-STD-016:F.2.3(3.3.1)]
    2. For each interface consider defining the following:
        (a) Priority that the interfacing entity(ies) is required to assign the interface [J-STD-016:F.2.3(3.x.a)]
        (b) Requirements on the type of interface (such as real-time data transfer, storage-and retrieval of data, etc.) to be implemented. [J-STD-016:F.2.3(3.x.b)]
        (c) Required characteristics of individual data elements that the interfacing entity(ies) will provide, store, send, access, receive, etc. [J-STD-016:F.2.3(3.x.c)] Consider the following when defining the requirements for the data element:
            (1) Names/identifiers and description of content.
            (2) Data type (alphanumeric, integer, etc.).
            (3) Size and format (such as length, bit-level description of data interface, and punctuation of a character string).
            (4) Units of measurement (such as meters, nanoseconds).
            (5) Range or enumeration of possible values (such as 0-99).
            (6) Precision (number of significant digits).
            (7) Priority, timing, frequency, volume, sequencing, and other constraints, such as whether the data element may be updated.
            (8) Sources (setting/sending entities) and recipients (using/receiving entities). [J-STD-016:F.2.3(3.x.c)]
        (d) Required characteristics of data element assemblies (records, messages, files, arrays, reports, etc.) that the interfacing entity(ies) will provide, store, send, access, receive, etc. [J-STD-016:F.2.3(3.x.d)] Consider the following when defining the requirements for the data element assembly:
            (1) Names/identifiers.
            (2) Data elements in the assembly and their structure (number, order, grouping).

(3) Relationships among assemblies, such as sorting/access characteristics.
(4) Priority, timing, frequency, volume, sequencing, and other constraints, such as whether the assembly may be updated.
(5) Sources (setting/sending entities) and recipients (using/receiving entities). [J-STD-016:G.2.2(3.x.d.1-8)]

(e) Identify the communication methods (e.g., Ethernet, 1553 Bus, S-Band Radio) that the interfacing entity(ies) is required to use for the interface. [J-STD-016:G.2.2(3.x.e)] The project may find value in including the following items:
(1) Communication links/bands/frequencies/media and their characteristics.
(2) Message formatting.
(3) Flow control (such as sequence numbering and buffer allocation).
(4) Data transfer rate, whether periodic/aperiodic, and interval between transfers.
(5) Routing, addressing, and naming conventions. [J-STD-016:G.2.2(3..x.e.1-8)]

(f) Required characteristics of protocols the interfacing entity(ies) will use for the interface (e.g., Transmission Control Protocol/ Internet Protocol (TCP/IP), User Datagram Protocol (UDP)). [J-STD-016:F.2.3(3.x.f)] The project may find value in including the following:
(1) Priority/layer of the protocol.
(2) Packeting, including fragmentation and reassembly, routing, and addressing.
(3) Legality checks, error control, and recovery procedures.
(4) Synchronization, including connection establishment, maintenance, termination.
(5) Status, identification, and any other reporting features. [J-STD-016:G.2.2(3.x.f.1-6)]

(g) Interface compatibility. This could be the identification of the transition mechanism/compatibility mode (e.g., hybrid dual Internet Protocol (IP) stack implementation to allow compatible communications between interfaces using IP Version 4 and IP version 6) or the identification of the subsets of functionality (to ensure compatibility between interfacing entities) in cases where both sides of the interface are using different versions of a standardized communication method/protocol. Other specifications, such as physical compatibility of the interfacing entity(ies) (dimensions, tolerances, loads, voltages, plug compatibility, etc.) may be included. [J-STD-016:G.2.2(3.x.g)]

e. [optional] Adaptation requirements (data used to adapt a program to a given installation site or to given conditions in its operational environment). [SWE-109.b.6]
f. [optional] Performance and timing requirements. [SWE-109.b.8] Specify required behavior of the software item and include applicable parameters, such as response times, throughput times, other timing constraints, sequencing, accuracy, capacities (how much/how many), priorities, continuous operation requirements, and allowable deviations based on operating conditions. Include, as applicable, required behavior under unexpected, unallowed, or “out of bounds” conditions, and requirements for error handling. [J-STD-016:F.2.4.(3.2.x)]
g. [optional] Security and privacy requirements. [SWE-109.b.9]
h. [optional] Environment requirements (e.g., computer hardware and operating system). [SWE-109.b.10]
i. [optional] Design and implementation constraints (e.g., requirements that constrain the design and construction of the system such as the use of a particular data standard, existing components, use of government-furnished property (equipment, information, or software), use of particular design or implementation standards, use of particular data standards, or programming language). [SWE-109.b.13]
j. [optional] Personnel-related requirements (e.g., specific requirements for each user type such as administrator and operator). [SWE-109.b.14]
k. [optional] Training-related requirements (e.g., on line help/tutorials). [SWE-109.b.15]
l. [optional] Packaging requirements (e.g., storage limitations on deliverable (on one disk), encryption, cyclic redundancy check on the deliverable, or digital signature). [SWE-109.b.17]
m. [optional] Testing requirements that drive software design decisions (e.g., checkpoint restart, the ability to see and modify variable values, monitoring execution time or bandwidth, the ability to inject defects). [SWE-109.f]
n. Bidirectional traceability between this document’s requirements and any higher level parent requirements documents. [SWE-109.d]

Appendix A4: Software Design Description

The Software Design Description shall include a design showing the following: [SWE-057] [SWE-111]

a. The decomposition into software items. [SWE_111.c] A software item is an aggregation of software, such as a computer program or database that satisfies an end use function and is designated for purposes of specification, testing, interfacing, configuration management, or other purposes. Software items are selected based on trade-offs among software function, size, host or target computers, developer, support strategy, plans for reuse, criticality, interface considerations, need to be separately documented and controlled, and other factors. [J-STD-016:3.1.37]
b. The interrelationship between software items. [SWE_111.c] Show the static (such as “consists of”) relationship(s) of the software items. Multiple relationships may be presented, depending on the selected software design methodology (for example, in an object-oriented design, this section may present the class and object structures as well as the module and process architectures of the software item). [J-STD-016:G.2.4(4.1.b0]
c. Concept of execution (e.g., a description or diagram that explains how the software items will interact during operation, initial conditions, response to each input, sequence of operations). [SWE_111.c.1.d] [J-STD-016:G.2.4(5.x.f.4)] This could include as applicable, flow of execution control, data flow, dynamically controlled sequencing, state transition diagrams, timing diagrams, priorities among software items, handling of interrupts, timing/sequencing relationships, exception and error handling, concurrent execution, dynamic allocation/deallocation, dynamic creation/deletion of objects, processes, tasks, and other aspects of dynamic behavior. [J-STD-016:G.2.4(4.20]
d. External interfaces, including Input/Output description. [SWE_111.c.1.b] [SWE_111.c.3] External entities include items such as systems, hardware items, software items, and users. If a software item contains, receives, or outputs data, include a description of its inputs, outputs, and other data elements and data element assemblies, as applicable. [J-STD-016:G.2.4(5)].
e. Traceability between the Software Requirements Specification and the Software Design Description.
f. Include the software design decisions (e.g., assumptions, limitations, and reliability related items/concerns or constraints) and rationale for those decisions. [SWE_111.a] [SWE_111.c.2]

Appendix A5: Software Test Plan (including test procedures)

The Software Test Plan shall include: [SWE-028] [SWE-013] [SWE-065] [SWE-104] [SWE-114]

a. Describe how the project will: [SWE-065.a]
   1. Document test inputs, [SWE-114.b.4]
   2. Record and evaluate test results, [SWE-030] [SWE-068] [SWE-069] [SWE-104.f]
   3. Document the evaluation, [SWE-068] [SWE-104.f]
   4. Address and track defects to closure, and [SWE-030] [SWE-069]
   5. Specify where results and evaluations are retained (e.g., a test log, defect tracking system, or matrix). [SWE-065.c]
b. Test milestones (e.g., the start and end dates for completing execution of the test plan; or the project may document these milestones in the SMP schedule). [SWE-028] [SWE 104.i]
c. Testing environment(s), site(s), personnel, and participating organizations. [SWE-104.k] [SWE-028] This section is completed when the project needs to plan in advance for a particular software test environment (e.g., the environment does not exist and it must be developed, environments needing the scheduling of particular resources, or specific configuration set up in order to run the tests). The testing environment includes software, hardware, and facilities necessary to run the tests, etc. Document the test resources and schedule for utilization of those resources (e.g., a lab or simulator or facility). Test preparation and configuration of the environment necessary to run the test can also be included here. [SWE-114.d]
d. Tests procedures. Generally include a test procedure identifier, the requirements addressed by the test, prerequisite conditions, test input, expected test results, criteria for evaluating results, and test procedure/operator actions. Testing order may also need to be specified. [SWE-028] [SWE-072] [SWE-065.b] [SWE-104.h]
   [SWE-114.b]
   1. Identify the requirement(s) that the test verifies and document the traceability between the test and the Software Requirements Specification (e.g., this can be done by constructing a traceability matrix between test cases and requirements). Multiple test cases may be needed to verify a single requirement. [SWE-028] [SWE-065.b] [SWE-072] [SWE-104.j] [SWE-114.b.2] [SWE-114.c]
   2. For each test describe the actions necessary to verify that the software fulfills the requirement(s). Actions may include establishing initial conditions, entering inputs/issuing commands, instructions for conducting the test, and evaluating the output against expected results and criteria. [SWE-028] [SWE-114.b] [SWE-072]

Appendix A6: Software Version Description

The Software Version Description shall contain: [SWE-063] [SWE_116]

a. Software name and the version identifier to which this Software Version Description applies (e.g., Software X –Version 9.2, Software X-MM/DD/YY, Software X- Release 2). [SWE_116.a] If there are multiple systems within the target environment(s) or the software configuration item is customized for a particular hardware configuration, determine if the system identification also needs to be included.
b. [optional] Summary of updates/changes since the previous Software Version Description, any open defects, and workarounds. [SWE_116.d] [SWE_116.h] [SWE_116.i] This could be a high-level summary of the enhancements and fixes or the project's current list of changes and defects and their status.
c. Instructions for building the executable software, including, for example, the instructions and data for compiling and linking and the procedures used for software recovery, software regeneration, testing, or modification. [SWE_116.e]
d. [optional] If the Software Version Description is not co-located with the software product files, include the location of the files. [SWE_116.g] Software product files are the set of computer programs, procedures, and associated documentation and data. Examples include requirements, design, source code, object code, databases, test information, batch files, command files, data files, manuals, and any files needed to install, build, operate, and maintain the software. [SWE_116.b] [SWE_116.g]