Software Assurance Objectives Hierarchy – Top Level

Top Objective: Software performs what is intended, only what is intended, and only in the intended manner

Strategy: Plan and execute Software Assurance throughout the software lifecycle

Objective: Software development and assurance processes are necessary and sufficient to achieve the project’s desired levels of safety, quality, security, and reliability

(1)

Objective: Software conforms to functional intent and performs as planned

(2)

Objective: Software does not adversely impact safety and contributes to system safety

(3)

Objective: Software system is robust and tolerant to failure & off nominal conditions

(4)

Objective: Software is secure and does not adversely impact safety and functionality of the system.

(5)

Objective: Software Verification and Validation Processes provide confidence in the interim and end products

(6)

SA Planning Section 9.1
SW Quality Section 9.2
SW Safety Section 9.3
SW Reliability & Maintainability Section 9.4
SW Security Section 9.5
SW V&V/IV&V Section 9.6/9.8
Objective: Software development and assurance processes are necessary and sufficient to achieve the project’s desired levels of safety, quality, security and reliability

1. SA Planning

- Strategy: Assure a software development process (1.A)
  - Objective: Software development planned processes have rigor appropriate to the project’s risk posture and conditions (1.A.1)
    - Strategy: Assess the software level and criticality (1.A.1.A)
    - Strategy: Assure a comprehensive and mature software development process is planned (1.A.1.B)
    - Strategy: Assure development organization(s) have the capability to perform as planned (1.A.1.C)
  - Strategy: Assure appropriate level SW development & assurance plans and processes are levied on providers (1.B.1.A)
  - Strategy: Assure acquired SW meets the project’s risk posture and conditions (1.B.1.B)

- Strategy: Assure a software acquisition process (1.B)
  - Objective: Software acquisition planned processes have rigor appropriate to the project’s risk posture and conditions (1.B.1)
    - Strategy: Estimate the cost of each applicable software assurance activity and the risk reduction it would provide (1.C.1.A)
    - Strategy: Create the software assurance plan (1.C.1.B)
    - Strategy: Allocate assurance personnel sufficient to execute SA plan (1.C.2.A)
    - Strategy: Train assurance personnel for needed skills & knowledge of project (1.C.2.B)
    - Strategy: Provide assurance personnel needed tools and resources (1.C.2.C)
    - Strategy: Provide SA access to project data & activities (1.C.2.D)

- Strategy: Plan software assurance activities (1.C)
  - Objective: Software assurance planned activities are appropriate to the project’s risk posture and conditions (1.C.1)
  - Objective: Software assurance is supported to the appropriate level (1.C.2)
Objective: Software conforms to functional intent and performs as planned (2)

Strategy: Achieve a high level of process maturity to ensure a robust software product (2.A)

Objective: Software Assurance processes provide reduced risks and higher confidence in SW products (2.A.1)

Strategy: Assess the software level and criticality and determine SA program risks (2.A.1.A)

Strategy: Provide a planned, maintained, comprehensive, working software assurance process (2.A.1.B)

Objective: Software development process minimizes insertion of errors (2.A.2)

Strategy: Assure use of a maintained, comprehensive working software development process (2.A.2.A)

Strategy: Assure software development processes are followed and do not introduce errors (2.A.2.B)

Objective: Software interim and final products conform to project needs and requirements (2.A.3)

Strategy: Assure software final and interim products are of sufficient quality for the project (2.A.3.A)

Strategy: Track, address, and trend issues via a closed loop problem resolution process (2.B.1.A)

Objective: Faults, defects, or other issues have been found and resolved as part of the development process (2.B.1)

Strategy: Track, address, and trend issues via a closed loop problem resolution process (2.B.1.A)

Objective: Remaining or known issues have been closed out to an acceptable level of risk (2.B.2)

Strategy: Identify and resolve faults throughout the development process in a timely manner (2.B)

Strategy: Assure risks are collected, analyzed, tracked, and addressed (2.B.2.A)

Context: Applies to full lifecycle from systems requirements to retirement
Objective: Software does not adversely impact safety and contributes to system safety (3)

Strategy: Ensure a complete understanding of all operational states of the system and the environment in which it will operate (3.A)

Objective: Software correctly reflects and responds to environmental and system states (3.A.1)

Strategy: Work with systems engineering and hardware to ensure the software reflects the system correctly (3.A.1.A)

Strategy: Assure the accuracy and fidelity of software used to simulate, model, and verify critical functions of the system (3.A.1.B)

Strategy: Create, follow and maintain project software safety plans and procedures based on project, SSCA and NS-8719.13 (3.B.1.B)

Strategy: Perform SW Safety Criticality Analyses (SSCA) on each project and record results (3.B.1.A)

Strategy: Provide a planned, maintained, comprehensive, working software safety process (3.B)

Strategy: Create, follow and maintain project software safety plans and procedures based on project, SSCA and NS-8719.13 (3.B.1.B)

Strategy: Assure software safety, system safety, reliability, project management, and SW engineering are coordinated & addressing safety (3.B.1.C)

Strategy: Software correctly reflects and responds to environmental and system states (3.A.1)

Strategy: Software correctly reflects and responds to environmental and system states (3.A.1)

Strategy: Work with systems engineering and hardware to ensure the software reflects the system correctly (3.A.1.A)

Strategy: Assure the accuracy and fidelity of software used to simulate, model, and verify critical functions of the system (3.A.1.B)

Strategy: Provide a planned, maintained, comprehensive, working software safety process (3.B)

Strategy: Software safety processes find, correct, control, mitigate, track, report and record effective SW Safety efforts (3.B.1)

Strategy: Identify all software contributions to potential hazards and eliminate, mitigate or control them to an acceptable level (3.C)

Strategy: Identify all software contributions to potential hazards and eliminate, mitigate or control them to an acceptable level (3.C)

Strategy: Identify & track critical system functions that software controls or impacts (3.C.2.A)

Strategy: Assure software controlled system hazards operate correctly (3.C.2.B)

Strategy: Provide a planned, maintained, comprehensive, working software safety process (3.B)

Strategy: Assure software contributions to hazards are identified, classified (SSCA performed), and documented to the appropriate level and reported to engineering in a timely manner (3.C.1.A)

Strategy: Identify software contributions to hazards are appropriately controlled, mitigated, or accepted and tracked to closure (3.C.1.B)

Strategy: Assure software contributions to hazards are appropriately controlled, mitigated, or accepted and tracked to closure (3.C.1.B)

Strategy: Assure SW design, implementation and testing address the control and mitigation of hazards caused or contributed to by software (3.C.1.C)

Strategy: Assure non-critical software does not impact safety-critical software (residing on same system, etc.) (3.C.1.D)

Strategy: Assure software safety features are present, sufficient, verified, tested and documented, and proof provided (3.C.1.E/3.C.2.C)
**Objective:** Software system is robust and tolerant to failure & off nominal conditions

**Strategy:** Assure that software is developed in a robust manner, which decreases/eliminates errors and determines residual risk (4.A)

**Objective:** Areas of software development weakness are known and addressed to the proper level (4.A.1)

**Strategy:** Assure appropriate software/system functional redundancy (4.B.1.D)

**Strategy:** Participate in system analysis and design to determine areas of weakness, suggesting where software design can improve system reliability (4.B.1.A)

**Strategy:** Predict remaining software faults based on testing and analyses, and address and report (4.A.2.B)

**Strategy:** Design in appropriate software architecture to meet and maintain critical functions (4.B.1.C)

**Strategy:** Determine appropriate level for functional redundancy between system and software (4.B.1.B)

**Strategy:** Design in protective barriers to prevent software from propagating system faults (system includes software, hardware, and human interface) (4.B.2.B)

**Strategy:** Assure appropriate software/system functional redundancy (4.B.1.D)

**Strategy:** Design in appropriate software architecture to meet and maintain critical functions (4.B.1.C)

**Strategy:** Analyze software functionality, requirements and design for vulnerabilities and recommend design strategies (4.B.2.A)

**Strategy:** Assure continued safe and reliable operations with managed and assured updates (4.C)

**Strategy:** Review and assess SW engineering’s maintainability plan assuring provisions for contracted, in-house, COTS, GOTS, Open Source and PLD version changes are addressed and followed (4.C.1.A)

**Strategy:** Assess and report on SW maintainability requirements, design and testing, checking for required system stated maintainability goals (4.C.1.B)

**Objective:** SW is designed to enhance system reliability (4.B.1)

**Objective:** SW is designed for robust operation (4.B.2)

**Objective:** SW is maintained for robust operation (4.C.1)

**Objective:** SW is designed for robust operation (4.B.2)

**Strategy:** Provide fault management (detection, isolation, recovery) capabilities (4.B.2.C)

**Strategy:** Analyze software functionality, requirements and design for vulnerabilities and recommend design strategies (4.B.2.A)

**Strategy:** Design in appropriate software architecture to meet and maintain critical functions (4.B.1.C)

**Strategy:** Determine appropriate level for functional redundancy between system and software (4.B.1.B)

**Strategy:** Design in protective barriers to prevent software from propagating system faults (system includes software, hardware, and human interface) (4.B.2.B)

**Strategy:** Assure appropriate software/system functional redundancy (4.B.1.D)

**Strategy:** Participate in system analysis and design to determine areas of weakness, suggesting where software design can improve system reliability (4.B.1.A)

**Strategy:** Predict remaining software faults based on testing and analyses, and address and report (4.A.2.B)

**Strategy:** Design in appropriate software architecture to meet and maintain critical functions (4.B.1.C)

**Strategy:** Determine appropriate level for functional redundancy between system and software (4.B.1.B)

**Strategy:** Design in protective barriers to prevent software from propagating system faults (system includes software, hardware, and human interface) (4.B.2.B)

**Strategy:** Assure appropriate software/system functional redundancy (4.B.1.D)

**Strategy:** Participate in system analysis and design to determine areas of weakness, suggesting where software design can improve system reliability (4.B.1.A)

**Strategy:** Predict remaining software faults based on testing and analyses, and address and report (4.A.2.B)

**Strategy:** Design in protective barriers to prevent software from propagating system faults (system includes software, hardware, and human interface) (4.B.2.B)

**Strategy:** Assure appropriate software/system functional redundancy (4.B.1.D)

**Strategy:** Participate in system analysis and design to determine areas of weakness, suggesting where software design can improve system reliability (4.B.1.A)

**Strategy:** Predict remaining software faults based on testing and analyses, and address and report (4.A.2.B)

**Strategy:** Design in protective barriers to prevent software from propagating system faults (system includes software, hardware, and human interface) (4.B.2.B)

**Strategy:** Assure appropriate software/system functional redundancy (4.B.1.D)
Objective: Software is secure and does not adversely impact safety and functionality of the system (5)

Strategy: Assure a secure development environment (5.A)

Objective: Software development environment, practices, and connectivity create secure development environment (5.A.1)

- Strategy: Assure SW Supply chain of environment equipment, tools, software (5.A.1.A)

Strategy: Assure secure Software practices (5.B)

Objective: Software development follows current practices to eliminate vulnerabilities and weaknesses (5.B.1)

- Strategy: Assure cyber security portion of Project Protection Plan is followed (5.B.1.A)
- Strategy: Assure usage of cyber best practices including vulnerability analyses tools (5.B.1.B)
- Strategy: Assure SW Supply chain for incorporated Open Source, COTS, OTS, PLDs, contracted software (5.B.1.C)

Strategy: Assure Operational security of software (5.C)

Objective: Software control of critical functions is correctly designed, implemented and secure (5.C.1)

- Strategy: Identify critical system functions and ensure software operates securely (5.C.1.A)
- Strategy: Software used to communicate operates correctly and securely (5.C.1.B)
- Strategy: Assure security evaluation of all updates to software, equipment, and usage (5.C.1.C)

Strategy: Assure SW Supply chain for incorporated Open Source, COTS, OTS, PLDs, contracted software (5.A.1.C)
Objective: Software Verification and Validation Processes provide confidence in the interim and end products (6)

Context: V&V includes reviews, inspections, testing, analyses, demos

**Strategy: Verify and validate functionality of software products (6.A)**

**Objective: Software V&V assures confidence in the interim and end software products (6.A.1)**

- Strategy: Perform reviews on software products, reporting results and findings to Project (6.A.1.A)
- Strategy: Track all review, inspection, testing findings to closure and monitor closure rate (6.A.1.B)
- Strategy: Assure test procedures, test sets, test drivers & stubs are complete properly run, and configuration managed; the test results are properly signed off (6.A.1.C)
- Strategy: Assure simulations and models are correct and provide necessary inputs & outputs (6.A.1.D)

**Strategy: Assure the software V&V processes are in place and working to find and remove defects (6.B)**

- Strategy: Independently verify and validate functionality and process for NASA’s most mission and safety critical software products (6.C)

**Objective: Assure Software V&V processes are providing robust and safe software products (6.B.1)**

- Strategy: Independently verify and validate functionality and process for NASA’s most mission and safety critical software products (6.C)

**Objective: Provide assurance that the safety and mission-critical software will operate reliably and safely (6.C.1)**

- Strategy: Assure systems’ software will perform expected functionality (6.C.1.A)
- Strategy: Assure systems’ software will not perform unwanted/undesired actions (6.C.1.B)
- Strategy: Assure systems’ software responds as expected under adverse conditions (6.C.1.C)