

<b>Id</b>	<b>Title</b>	<b>Status</b>
M-1	Validate Interface Requirements by Inspection Against Component Interfaces	Active
M-2	Validate Requirements by Inspecting Against Quality Criteria and System/Software Background Artifacts	Active
M-3	Validate Requirements by Inspecting Bidirectional Traces	Active
M-4	Verify Implementation of Requirements or Design in Source Code or Scripts through Manual Inspection	Active
M-5	Determine Reuse Applicability by Manually Comparing Operational Environments	Active
M-6	Validate Safety Requirements by Inspection of Traces to Fault Trees and FMEA	Active
M-7	Validate Requirements by Inspecting Traces to Scenarios	Active
M-8	Verify Autogenerated Software Implementation by Inspecting Traces from Requirements to Rational Rose Design Model	Active
M-9	Verify Software Code Quality using Static Analysis Tools	Active
M-10	Validate Test Plan by Inspection	Active
M-11	Verify Test Execution by Inspection of Test Cases, Inputs and Results	Active
M-12	Verify Adequacy of Software Architecture Dynamically with Executable UML Models	Active
M-13	Verify SW Interface Implementation by Inspection Against Interface Design	Active
M-14	Verify Software Behavior for Off-Nominal Conditions using Independent Testing	Active
M-15	Verify and Validate Software Design using UML Model	Active
M-16	Validate Test Design by Inspecting Test Scripts and Validated Requirements	Active
M-17	Validate Software Architecture by Inspecting Traces to Essential Properties	Active
M-18	Validate Software Interface Design Dynamically using Executable UML	Active
M-19	Validate Requirements Dynamically with Executable UML Models	Active
M-20	Verify Critical Software Changes By Inspecting Change Requests	Active
M-21	Validate Test Design by Inspecting Traces from Scenarios	Active
M-22	Validate Test Environment by Inspecting Against Inputs and Conditions Identified from Behaviors/Scenarios	Active
M-23	Validate Test Cases Dynamically Using Independent Testing	Active
M-24	Validate Software Requirements by Tracing to Essential Properties of the System	Active
M-25	Validate Test Cases by Inspection and Traces to Requirements	Active
M-26	Validate Test Cases by Inspection Against Requirements and System Reference Model	Active
M-27	Verify Software Architecture and Performance Requirements Dynamically using MagicDraw™ Cameo	Active
M-28	Verify System/Software Architecture Using a Discrete Model of Performance Requirements in Stressing Scenarios	Active
M-29	Assess Architecture Completeness by Inspection Against an Architectural Standard	Active
M-30	Validate Software Requirements by Inspecting Traces to SysGoals	Active
M-31	Validate Mission Project Use Cases by Inspection Against Concept Documentation	Active
M-32	Verify Scripted Timeline Via Manual Multi-Directional Tracing	Active
M-33	Validate Key Driving Requirement by Tracing to System Architecture and Stakeholder Expectations	Active
M-34	Validate Feasibility Study Conclusions by Inspection	Active
M-35	Validate Test Procedures by Inspection and Traces to Requirements	Active
M-36	Validate Mission Project Operational Concepts by Generating Use Cases from Concept Documentation	Active
M-37	End-to-End Fault Management Verification through Database Development and Analysis	Active
M-38	Verify Software Implementation by Inspecting Traces to Requirements (Nominal, Off-Nominal and Hazards Scenarios)	Active
M-39	Verify Software Design by Inspecting Traces to Requirements and Software Architecture	Active
M-40	Validate System Behaviors Dynamically by Executing Simulations/Models	Active
M-41	Verify Software Interface Design by Inspection Against Interface Requirements	Active
M-42	Validate algorithm design through algorithm qualitative attribute assessment	Active
M-43	Verify Interface Implementation By Tracing Component Behaviors to Interface Requirements and Source Code To Identify Missing or Unnecessary Interface Requirements	Active
M-44	Verify Requirements Implemented using Off the Shelf Components through Structural Analysis to Evaluate Protections Against Execution of Dormant Code	Active
M-45	Verify Requirements Implemented using OTS Components By Tracing Requirements to OTS Functional Documentation to Identify Unnecessary Functions and Missing Capabilities	Active
M-46	Validate Complex Network Requirements, Architecture, Design, and Test Documentation through Layered Model Analysis	Active
M-47	Verify via dynamic testing that test artifacts fully cover implemented requirements	Active
M-48	Verifying Functionality/Behaviors Based on Static Analysis Results	Active
M-49	Verify via dynamic testing that reused software is free from on-orbit anomalies discovered on legacy software	Active
M-50	Validate candidate software issues via dynamic testing to confirm issue viability	Active
M-51	Validate Modeled System Designs by Employing the Model-Based Testing Function of RoseRT to Demonstrate Required Behaviors	Active
M-52	Validate System Security Categorization and Regulatory Security Requirements by Inspection using Security Risk Management Framework (NIST-SP-800-37, Step 1)	Active
M-53	Verify Security Control Selection and Threats/Risks Identification by Inspection using Security Risk Management Framework (NIST-SP-800-37, Step 2)	Active
M-54	Verify Security Control Implementation by Inspection using Security Risk Management Framework (NIST-SP-800-37, Step 3)	Active
M-55	Verify Security Remediation Actions by Inspection using Security Risk Management Framework (NIST-SP-800-37, Step 4)	Active
M-56	Verify Security Risk Action Plan before Operations by Inspection using Security Risk Management Framework (NIST-SP-800-37, Step 5)	Active
M-57	Verify System Software Safety by Comparing Concept Documentation, Requirements, Testing, Design and Code with Hazard Analysis Documentation to Establish a Safety Case, Across the Sc	Active
M-58	Examine System Design and Requirements using a Fault Tree Analysis Tool to Identify Potential Hazards and Their Causes	Active
M-59	Verify Interface Implementation in Software by Simulated Dynamic Testing to Demonstrate Successful Software Component Integration	Active
M-60	Verify Software Capabilities through Independent Testing of Operational Scenarios	Active
M-61	Verify Software Implementation by Executing Independent Regression Tests to Demonstrate Software Behaves as Expected	Active
M-62	Verify Suitability of Developer's Test Environment by Simulation of Developer Test Operations Against Test Environment Validation Criteria	Active
M-63	Verify Test Coverage of Software through Analysis of Source Code and Test Artifacts using Software Coverage Analysis Tools and Coverage Criteria	Active
M-64	Verify Performance Requirements Implementation via Simulated Dynamic Testing to Stress Software Boundaries and Limitations	Active
M-65	Validate and Verify Regression Test Scope and Changes to Artifacts to Establish Integrity by Inspection	Active
M-66	Validate Use Cases by Inspection Against Driving Documentation	Active
M-67	Verify Requirements Implementation By Analyzing Test Results Using Statechart Assertions	Active
M-68	Validate Key Capabilities via Dynamic Testing against High Risk Scenarios to Reduce Risk of Operations	Active
M-69	Plan a Test Approach to Meet IV&V Analysis Objectives of Project s IPEP /TS&R	Active
M-70	Validate System Interface Requirements by Inspection Against Documentation	Active
M-71	Verify Software Design by Inspecting Traces to Requirements Using MATLAB and Rhapsody	Active
M-72	Verify Software Implementation by Inspecting Traces to Requirements Using Understand for C++	Active
M-73	Validate Requirements by Inspecting Against Quality Criteria	Active
M-74	Scenario-Based Requirements Analysis	Active
M-75	Verify Flight Software Architecture With UML Model	Active
M-76	Verify SW Design	Active
M-77	Verify Requirements Implementation	Active
M-78	Test Case Analysis	Active
M-79	Stakeholder Analysis	Active
M-80	IV&V Real Time Anomaly Support (RTAS)	Active
M-81	Manual Source Code analysis (implementation of requirements/design)	Active
M-82	Verify Interface Design using UML Model	Active
M-83	Change Impact Analysis	Active
M-84	Verify Architecture by Dynamic Simulation	Active
M-85	Validate Interface Design through Model Checking	Active
M-86	Verify Design and Implementation	Active

<b>Id</b>	<b>Title</b>	<b>Status</b>
M-87	Validate Requirements	Active
M-88	Validate Test Artifacts per Build	Active
M-89	Artifact Delta Evaluation	Active
M-90	Verify Design and Implementation - Compound Method (Proposed)	Active
M-91	Verify FSW Design to Requirements	Active
M-92	Static Code Analysis	Active
M-93	Test Results Analysis	Active
M-94	Qualitative Assessment of Requirements	Active
M-95	Requirements Traceability Analysis (RTA)	Active
M-96	Validate Test Plan	Active
M-97	Verify Software Architecture Against Performance Requirements	Active
M-98	Scenario Based Software Implementation Verification	Active
M-99	Independent Testing	Active
M-100	Validate Mission Project Operational Concepts by Inspection Against Concept Documentation	Active
M-101	Verify Design Provides Dependability and Fault Tolerance	Active
M-102	Verify SW Interface Implementation by Inspection Against Interface Design - Retired	Active
M-103	Verify and Validate Requirement Implementation using Flow Diagrams to Uncover Missing, Conflicting, or Unnecessary Behavior	Active
M-104	Verify Planning Artifacts for Legacy to Replacement Systems Transitions via Comparison to Relevant System Migration Standards to Ensure Effective and Complete System Transition and Ret	Active
M-105	Verify PBRA/RBA Critical Behaviors are Appropriately Rated by Correlating Issues to Behaviors	Active
M-106	Validate Program to Program High Level Concept Documentation by Manual Evaluation of Alignment of Required and Provided Services Across Elements	Active
M-108	Validate Software Requirements meet dependability criteria and control identified hazards by inspection of traces to identified dependability attributes and safety hazards	Active
M-109	Verify System Software Safety Documentation Identifies all known software based hazard causes and controls by inspection of Adverse Conditions/Failure Modes	Active
M-110	Verifying CFS-related Fault Management Implementation in Source Code	Active
M-111	Verify Requirements by Tracing from Requirements to Adverse Conditions to Test Results to assure Requirements do not cause Hazardous Conditions Discovered in Testing	Active
M-112	Verify Subsystem Requirements and Architecture Elements to Address Stakeholder Needs and Manual Examination of the Relationships Between System Architecture, Subsystem Requirements	Active
M-113	Determine CSCI release content that satisfies mission needs and enhances safety by analyzing open Software Change Requests (SCR)	Active
M-114	Assess CSCI deployment readiness by analyzing mission needs, results of life cycle reviews, and impact of open Software Change Requests(SCR)	Active
M-115	Using Systems Tool Kit (STK) to Perform Analysis on Project Telemetry	Active
M-116	Uncover risk or assurance by using performance analysis techniques on mission critical COTS/GOTS software.	Active
M-117	Validate Software Fault Handling by Independent Testing with Simulations	Active
M-118	Verify performance requirements by monitoring system impacts of mission critical COTS/GOTS software.	Active
M-119	Verify Implementation by Tracing Data Persistence Code to Values Used in Embedded SQL Code	Active
M-120	Blue Team Vulnerability Assessment Method: Pre-Assessment (Phase 1)	Active
M-121	Blue Team Vulnerability Assessment Method: Active-Assessment (Phase 2)	Active
M-122	Blue Team Vulnerability Assessment Method: Post-Assessment (Phase 3)	Active
M-123	Assure Cross-System Integrity of Critical Commands and Data	Active
M-124	Develop Independent Capability-based Scenarios from Design Documentation	Active
M-125	Validate Fault Protection Design and Requirement Flowdown	Active
M-126	Verify JUnit implementation through manual analysis.	Active
M-127	Assess Adequacy of Unit Tests	Active