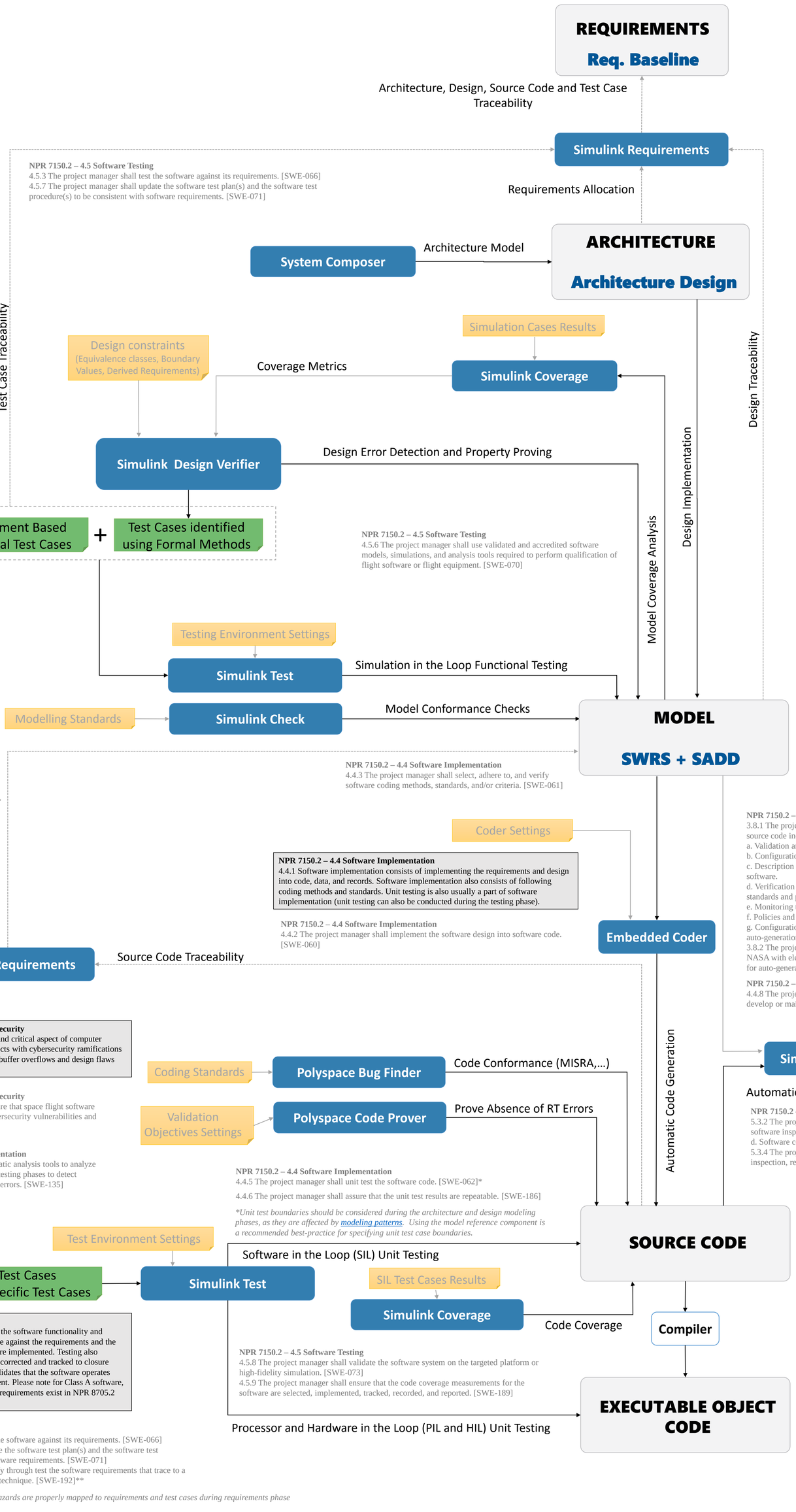
MathWorks®

Accelerating the pace of engineering and science

NASA NPR 7150.2C Compliant Flight Software Development Workflow



NPR 7150.2 – 4.1 Software Requirements
4.1.1 The requirements phase is one of the most critical phases of software engineering. Studies show that the top problems in the software industry are due to poor requirements elicitation, inadequate requirements specification, and inadequate management of changes to requirements. Requirements provide the foundation for the entire life cycle, as well as for the software product. Requirements also provide a basis for planning, estimating, and monitoring. Requirements are based on customer, user, and other stakeholder needs and design and development constraints. The development of requirements includes elicitation, analysis, documentation, verification, and validation. Ongoing customer validation of the requirements to ensure the end products meet customer needs is an integral part of the life-cycle process. Customer validation can be accomplished via rapid prototyping and customer-involved reviews of iterative and final software requirements.

NPR 7150.2 – 4.1 Software Requirements
4.1.2 The project manager shall establish, capture, record, approve, and maintain software requirements, including requirements for COTS, GOTS, MOTS, OSS or reused software components, as part of the technical specification. [SWE-050]
4.1.3 The project manager shall perform software requirements analysis based on flowed-down and derived requirements from the top-level systems engineering requirements, safety and reliability analyses, and the hardware specifications and design. [SWE-051]
4.1.3.1 The project manager shall track and manage changes to the software requirements. [SWE-053]

NPR 7150.2 – 3.12 Software Bi-Directional Traceability

3.12.1 The project manager shall perform, record, and maintain bi-directional traceability between the following software elements: [SWE-052]

Table 1. Bi-directional traceability by software classification

Bi-directional Traceability	Class A, B, and C	Class D	Class F
Higher-level requirements to the software requirements	X		X
Software requirements to the system hazards	X	X	
Software requirements to the software design components	X		
Software design components to the software code	X		
Software requirements to the software test procedures	X	X	X
Software requirements to the software non-conformances	X	X	X

NPR 7150.2 – 4.2 Software Architecture
4.2.1 Experience confirms that the quality and longevity of a software-reliant system is primarily determined by its architecture. The software architecture underpins a system's software design and code; it represents the earliest design decisions, ones that are difficult and costly to change later. The transformation of the derived and allocated requirements into the software architecture results in the basis for all software development work.

NPR 7150.2 – 4.2 Software Architecture
4.2.3 The project manager shall transform the requirements for the software into a recorded software architecture. [SWE-057]
4.2.4 The project manager shall perform a software architecture review on the following categories of projects: [SWE-143]
a. Category 1 Projects as defined in NPR 7120.5.
b. Category 2 Projects as defined in NPR 7120.5 that have Class A or Class B payload risk classification per NPR 8705.4.

NPR 7150.2 – 4.3 Software Design
4.3.1 Software design is the process of defining the software architecture, components, modules, interfaces, and data for a software system to satisfy specified requirements. The software architecture is the fundamental organization of a system embodied in its components, their relationships to each other and the environment, and the principles guiding its design and evolution. The software architectural design is concerned with creating a strong overall structure for software entities that fulfill the allocated system and software-level requirements. Typical views captured in an architectural design include the decomposition of the software subsystem into design entities, computer software configuration items, definitions of external and internal interfaces, dependency relationships among entities and system resources, and finite state machines. The design should be further refined into lower-level entities that permit the implementation by coding in a programming language. Typical attributes that are documented for lower-level entities include the identifier, type, purpose, function, constraints, subordinates, dependencies, interface, resources, processing, and data. Rigorous specification languages, graphical representations, and related tools have been developed to support the evaluation of critical properties at the design level. Projects are encouraged to take advantage of these improved design techniques to prevent and eliminate errors as early in the life-cycle as possible.

NPR 7150.2 – 4.3 Software Design
4.3.2 The project manager shall develop, record, and maintain a software design based on the software architectural design that describes the lower-level units so that they can be coded, compiled, and tested. [SWE-058]

- Software Development Plan (SDP)
  - Modeling and Design Standards
- SW Requirements Specification (SRS)
  - Software Logical Model
  - Req. Traceability Matrix
- SW Design Document (SDD)
  - Software Architectural Design
  - Software Behaviour
  - Internal Interface Design
- Software Verification Report (SVR)
  - Autocode input model review
  - Test Traceability Matrix
  - Model Conformance Report
  - Model Coverage Report

...

NPR 7150.2 – 3.8 Automatic Generation of Software Source Code
3.8.1 The project manager shall define the approach to the automatic generation of software source code including: [SWE-146]
a. Validation and verification of auto-generation tools.
b. Configuration management of the auto-generation tools and associated data.
c. Description of the limits and the allowable scope for the use of the auto-generated software.
d. Verification and validation of auto-generated source code using the same software standards and processes as hand-generated code.
e. Monitoring the actual use of auto-generated source code compared to the planned use.
f. Policies and procedures for making manual changes to auto-generated source code.
g. Configuration management of the input to the auto-generation tool, the output of the auto-generation tool, and modifications made to the output of the auto-generation tools.
3.8.2 The project manager shall require the software developers and suppliers to provide NASA with electronic access to the models, simulations, and associated data used as inputs for auto-generation of software. [SWE-206]

NPR 7150.2 – 4.4 Software Implementation
4.4.8 The project manager shall validate and accredit software tool(s) required to develop or maintain software. [SWE-136]

NPR 7150.2 – 4.4 Software Implementation
4.4.1 Software implementation consists of implementing the requirements and design into code, data, and records. Software implementation also consists of following coding methods and standards. Unit testing is also usually a part of software implementation (unit testing can also be conducted during the testing phase).

NPR 7150.2 – 4.4 Software Implementation
4.4.2 The project manager shall implement the software design into software code. [SWE-060]

NPR 7150.2 – 4.4 Software Implementation
4.4.3 The project manager shall select, adhere to, and verify software coding methods, standards, and/or criteria. [SWE-061]

NPR 7150.2 – 4.5 Software Testing
4.5.6 The project manager shall use validated and accredited software models, simulations, and analysis tools required to perform qualification of flight software or flight equipment. [SWE-070]

NPR 7150.2 – 4.5 Software Testing
4.5.3 The project manager shall test the software against its requirements. [SWE-066]
4.5.7 The project manager shall update the software test plan(s) and the software test procedure(s) to be consistent with software requirements. [SWE-071]

NPR 7150.2 – 3.11 Software Cybersecurity
3.11.1 Software defects are a central and critical aspect of computer security vulnerabilities. Software defects with cybersecurity ramifications include implementation bugs such as buffer overflows and design flaws such as inconsistent error handling.

NPR 7150.2 – 3.11 Software Cybersecurity
3.11.5 The project manager shall ensure that space flight software systems are assessed for possible cybersecurity vulnerabilities and weaknesses. [SWE-158]

NPR 7150.2 – 4.4 Software Implementation
4.4.4 The project manager shall use static analysis tools to analyze the code during the development and testing phases to detect defects, software security, and coding errors. [SWE-135]

NPR 7150.2 – 4.4 Software Implementation
4.4.5 The project manager shall unit test the software code. [SWE-062]*
4.4.6 The project manager shall assure that the unit test results are repeatable. [SWE-186]

**Unit test boundaries should be considered during the architecture and design modeling phases, as they are affected by modeling patterns. Using the model reference component is a recommended best-practice for specifying unit test case boundaries.*

NPR 7150.2 – 4.5 Software Testing
4.5.1 The purpose of testing is to verify the software functionality and remove defects. Testing verifies the code against the requirements and the design to ensure that the requirements are implemented. Testing also identifies problems and defects that are corrected and tracked to closure before product delivery. Testing also validates that the software operates appropriately in the intended environment. Please note for Class A software, additional software test and integration requirements exist in NPR 8705.2 beyond those listed below.

NPR 7150.2 – 4.5 Software Testing
4.5.3 The project manager shall test the software against its requirements. [SWE-066]
4.5.7 The project manager shall update the software test plan(s) and the software test procedure(s) to be consistent with software requirements. [SWE-071]
4.5.12 The project manager shall verify through test the software requirements that trace to a hazardous event, cause, or mitigation technique. [SWE-192]**

***Satisfying SWE-192 requires that hazards are properly mapped to requirements and test cases during requirements phase*

NPR 7150.2 – 4.5 Software Testing
4.5.8 The project manager shall validate the software system on the targeted platform or high-fidelity simulation. [SWE-073]
4.5.9 The project manager shall ensure that the code coverage measurements for the software are selected, implemented, tracked, recorded, and reported. [SWE-189]

NPR 7150.2 – 4.4 Software Implementation
4.4.7 The project manager shall provide a software version description for each software release. [SWE-063]

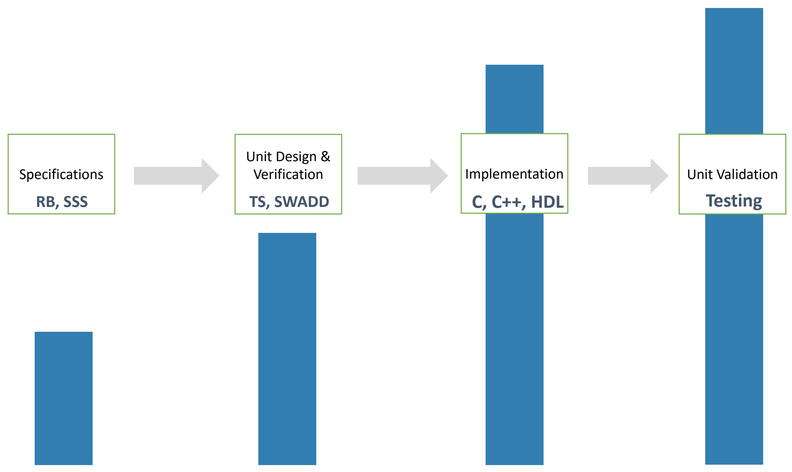
NPR 7150.2 – 4.5 Software Testing
4.5.2 The project manager shall establish and maintain: [SWE-065]
a. Software test plan(s).
b. Software test procedure(s).
c. Software test report(s).
4.5.5 The project manager shall evaluate test results and record the evaluation. [SWE-068]

NPR 7150.2 – 4.5 Software Operations, Maintenance, and Retirement
4.6.3 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. [SWE-077]

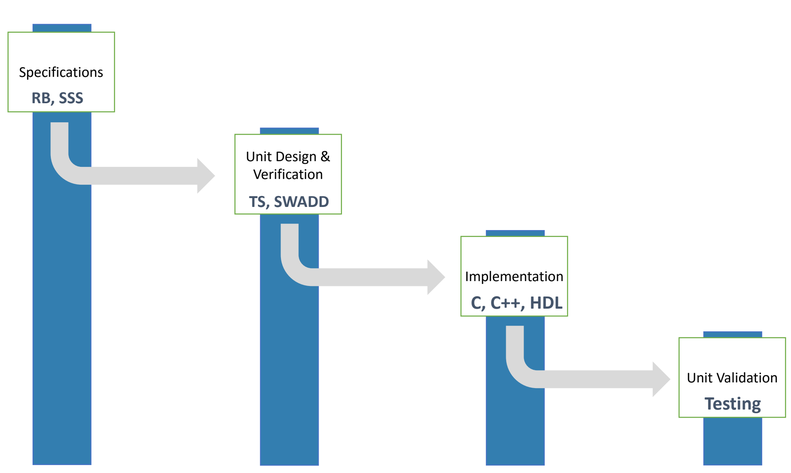
- Software Development Plan (SDP)
  - Coding standards
  - Test Traceability Matrix
  - Tool Validation Documentation
- Software Unit Integration Test Plan (SUITP)
- Software Validation Report (SVR)
  - Code Generation Report
  - Software Unit Test Report
  - Software Code Traceability Matrix
  - Code Coverage Report
  - Robustness Report
  - Independent SW Validation Report
  - Review and Inspection Reports

...

Effort Distribution in Traditional Development Workflows



Effort Distribution in Model-Based Design Workflows



MATLAB® & SIMULINK®

Prepared by: Ossi Saarela, MathWorks
osaarela@mathworks.com
+1-508-647-1618

December 2019