1. **Purpose**

This document defines the policies and procedures utilized for developing, verifying, and releasing software associated products. It defines the Waterfall development process used and specifies the activities associated with software planning, requirement, architecture, detailed design, unit testing, integration and system testing, and release.

1. **Scope**

This process applies to all software incorporated into distributed medical devices. Any software systems not intended for commercialization or the quality management system are exempt from this process.

1. **General** 
   1. **Definitions**

**Software of Unknown Provenance (SOUP)** – Software item that is already development and generally available and that has not been developed for the purpose of being incorporated into the medical device (also known as “off-the-shelf software”) or software previously development for which adequate records of the development processes are not available.

* 1. **Responsibilities**

**Engineering** – Engineering is responsible for owning this process and ensuring each step is completed and documented.

**Quality Management** – Quality Management is responsible for the implementation and continued compliance with the policies and procedures specified in this document and by the regulatory authorities.

* 1. **Equipment and Materials** – N/A
  2. **Safety Precautions** – N/A
  3. **Training Requirements** – All Engineering and Quality personnel shall be trained to this procedure and the training documented.
  4. **Record Management** – All records associated with this document shall be maintained within the associated Design History File.
  5. **Reference Documents and Materials**

**ANSI/AAMI/IEC 62304** – Medical Device Software – Software Life Cycle Processes

**Guidance for Industry and FDA Staff** – General Principles of Software Validation

**QP-0009** – Change Control Process

1. **Software Development Process Overview**

The organization utilizes the following development and supporting procedures to ensure a robust and controlled software development process. The level of detail associated with each step of the process is commensurate with the risk associated with the software.

| Software development processes:   * Development Planning * Requirement Analysis and Traceability * Architecture Design * Detailed Design * Unit Implementation and Verification * Integration and System Testing * Final Validation Report | Supporting processes:   * Risk Management * Software Configuration Management * Software Problem Resolution |
| --- | --- |



1. **Software Development Planning**

The objective of development planning is to reduce risks caused by software, communicate procedures and goals to team members, and ensure quality requirements for the software are met. The planning should be specified at the level of detail necessary to carry out the development process and should be proportional to the associated risk. Planning is an iterative activity that shall be re-evaluated and updated as development progresses as necessary.

* 1. **Software Development Plan**

The development plan shall list the activities necessary to complete the software development process based on the scope, magnitude, and safety classification of the software being developed. The following items shall be addressed in the plan:

* The process to be used in the development of the software
* The deliverables, including documentation, of the activities and tasks
* Traceability between system requirements, software requirement specifications, software system testing, and risk control measures
* Software configuration and change management, including SOUP software and software used to support development
* Software problem resolution for handling problems detected in the software products, deliverables, and activities at each stage of the life cycle
* Review the FDA Guidance – Deciding When to Submit a 510(k) for a Change to an Existing Device
  1. **Software Development Plan reference to System Design and Development**

The development plan shall reference system requirements as inputs for software development. The plan shall include or reference procedures for coordinating the software development and the design and development validation.

* 1. **Software Development Standards, Methods, and Tools Planning**

The development plan shall include or reference the standards, methods, and tools utilized in the development process.

* 1. **Software Integration and Integration Testing Planning**

The development plan shall account for the integration of the software items and the testing necessary to ensure effective integration. This may be combined with Software System testing into a single plan and set of activities.

* 1. **Software Verification Planning**

The following information shall be within the software development plan:

* + - * Deliverables requiring verification
      * The required verification tasks for each life cycle activity
      * Milestones at which the deliverables are verified
      * The acceptance criteria for verification of the deliverables
  1. **Software Risk Management Planning**

The development plan shall include a plan to conduct the activities and tasks required by risk management, including the management of risks relating to SOUP.

* 1. **Software Safety Classification**

The plan shall include a safety class based on possible hazards to users. The categories are dependent on severity as follows:

* Class A – No injury or damage to health is possible
* Class B – Non-serious injury is possible
* Class C – Death or serious injury is possible
  1. **Documentation Planning**

The software development plan shall include information regarding the documents to be produced or amended during the software development life cycle.

* 1. **Software Configuration Management Planning**

The following software configuration management information shall be included in the development plan:

* + - * The classes, types, categories, or lists of items to be controlled
      * The software configuration management activities and tasks
      * The organization responsible for performing software configuration management and activities
      * Their relationship with other organizations, such as software development or maintenance
      * When the items are to be placed under configuration control
      * When the problem resolution process is to be used
  1. **Support Items to be Controlled**

The development plan shall include any development tools, items or settings, which could impact the software.

* 1. **Software Configuration Item Control before Verification**

The manufacturer shall plan to place configuration items under documented configuration management control before they are verified.

1. **Software Requirements Analysis**

The objective of requirement specifications is to establish verifiable requirements that define what is to be built, demonstrate the software exhibits the required behavior, and ensure the software is complete and ready for use. It is essential that requirements be stated in a manner that objective criteria can be obtained to verify it was properly implemented. Any risk management requirements necessary to mitigate risk or regulatory conformance requirements shall be identified in the software requirements.

* 1. **Define and Document Software Requirements from System Requirements**

For each software system of the medical device, define and document software system requirements from the system level requirements

* 1. **Software Requirement Content**

Software requirements shall contain the following requirement as appropriate to the medical device:

* Functional and capability requirements
* Software system inputs and outputs
* Interfaces between the software system and other systems
* Software-driven alarms, warnings, and operator messages
* Security requirements
* Usability engineering requirements that are sensitive to human errors and training
* Data definition and database requirements
* Installation and acceptance requirements of the delivered medical device software at the operation and maintenance site or sites
* Requirements related to methods of operation and maintenance
* User documentation to be developed
* User maintenance requirements
* Regulatory requirements
  1. **Include Risk Control Measures in Software Requirements**

Risk control measures implemented in software for hardware failures and potential software defects shall be included in the requirements as appropriate.

* 1. **Re-Evaluate Medical Device Risk Analysis**

The medical device risk analysis shall be re-evaluated when the software requirements are established and updated as necessary.

* 1. **Update System Requirements**

All requirements, including system requirements, shall be re-evaluated and updated as appropriate throughout the software development process.

* 1. **Verify Software Requirements**

The software requirements shall be verified to meet the following conditions:

* Implement system requirements including those relating to risk control
* Do not contradict one another
* Are expressed in terms that avoid ambiguity
* Are stated in terms that permit establishment of criteria and performance of tests to determine whether the test criteria have been met
* Can be uniquely identified
* Are traceable to system requirements or other source

1. **Architecture Development**

The objective of the architecture development is to define the major structural components of the software, their externally visible properties, and the relationship between them. All interrelated component behaviors shall be described in the software architecture. This activity is not complete until all software requirements can be implemented by the defined software items. The software architecture provides the foundation for the detailed design and implementation of the software. The risk classification shall be re-evaluated upon completion of the architecture design.

* 1. **Transform Software Requirements into an Architecture**

The requirements for the medical device software shall be transformed into a documented architecture that describes the software’s structure and identifies the software items.

* 1. **Develop an Architecture for the Interfaces of Software Items**

An architecture shall be developed and documented for the interfaces between the software items and the components external to the software items (software and hardware), and between the software items.

* 1. **Specify Functional and Performance Requirement of SOUP Items**

For any software identified as SOUP, the functional and performance requirements that are necessary for its intended use shall be specified for the SOUP item.

* 1. **Specify System Hardware and Software Required by SOUP Items**

For any software identified as SOUP, the system hardware and software necessary to support the proper operation of the SOUP items shall be specified.

* 1. **Identify Segregation Necessary for Risk Control**

Any segregation between software items that is essential to risk control shall be identified and stated how to ensure that the segregation is effective.

* 1. **Verify Software Architecture**

The software architecture shall be verified to contain the following:

* The architecture of the software implements system and software requirements including those relating to risk control
* The software architecture is able to support interfaces between software items and between software items and hardware
* The medical device architecture supports proper operation of any SOUP items

1. **Detailed Design**

The objective of the detailed design is to refine the software items and interfaces defined in the architecture to create software units and their interfaces that can be tested separately. The detailed design specifies algorithms, data representations, interfaces among different software units, and interfaces between software units and data structures. The design provides the necessary details to construct the software and should be complete enough as to not require the programmer to made ad hoc design decisions.

* 1. **Refine Software Architecture into Software Units**

The software architecture shall be refined until it is represented by software units.

* 1. **Develop Detailed Design for each Software Unit**

A detailed design for each software unit of the software item shall be developed and documented.

* 1. **Develop Detailed Design for Interfaces**

A detailed design for any interfaces between the software unit and external components (hardware or software) shall be developed and documented, as well as any interfaces between software units.

* 1. **Verify Detailed Design**

The detailed design shall be verified that the following items are satisfied:

* Implements the software architecture
* Is free from contradiction with the software architecture

1. **Software Unit Implementation and Verification**

The objective of this activity is to verify the translation of the software architecture and detailed design into source code. The source code for each software unit shall be verified to ensure that it functions as specified by the detailed design.

* 1. **Implement each software unit**

Each software unit shall be implemented.

* 1. **Establish Software Unit Verification Process**

Strategies, methods, and procedures for verifying each software unit shall be established. Where verification is done by testing, the test procedures shall be evaluated for correctness.

* 1. **Software Unit Acceptance Criteria**

Acceptance criteria for software units prior to integration into larger software items shall be established as appropriate, and ensured that software units meet acceptance criteria.

* 1. **Additional Software Unit Acceptance Criteria**

When present in the design, additional acceptance criteria shall be included for:

* Proper event sequence
* Data and control flow
* Planned resource allocation
* Fault handling (error definition, isolation, and recovery)
* Initialization of variables
* Self-diagnostics
* Memory management and memory overflows
* Boundary conditions
  1. **Software Unit Verification**

Software unit verification shall be performed and the results documented.

1. **Software Integration and Integration Testing**

The objective of software integration and integration testing to execute the integration of software units into aggregate software items and verify the resulting software items behave as intended. The approach to integration is dependent on the software items being assembled. As applicable, integration testing shall demonstrate program behavior at the boundaries of its inputs/output domains and confirm program responses to invalid, unexpected, and special inputs.

* 1. **Integrate Software Units**

The software unit shall be integrated in accordance with the integration plan.

* 1. **Verify Software Integration**

The following aspects of the software integration in accordance with the integration plan shall be verified and recorded:

* The software units have been integrated into software items and the software system
* The hardware items, software items, and support for manual operations (e.g., human-equipment interface, on-line help menus, speech recognition) of the system have been integrated into the system
  1. **Test Integrated Software**

The integrated software items shall be tested in accordance with the integration plan and the results documented.

* 1. **Integration Testing Content**

For integration testing, whether the integrated software item performs as intended shall be addressed.

* 1. **Verify Integration Test Procedures**

The integration test procedures shall be evaluated for correctness.

* 1. **Conduct Regression Tests**

When software items are integrated, regression testing shall be conducted as appropriate to demonstrate that defects have not been introduced into previously integrated software.

* 1. **Integration Test Record Contents**

Integration test records shall contain:

* Documentation of the test result (pass/fail and a list of anomalies)
* Sufficient records to permit the test to be repeated
* The identity of the tester
  1. **Use Software Problem Resolution Process**

Anomalies found during software integration and integration testing shall be entered into a software problem resolution process.

1. **Software System Testing**

The objective of software system testing is to verify the software’s functionality by demonstrating that the requirements for the software have been successfully implemented. Tests for each requirement can be completed individually or by combinations of requirements.

Software system testing may be performed in a simulated environment, actual target hardware, or on the full medical device as appropriate. When a change is made to a software system, the degree of regression testing (not just testing of the individual change) should be determined to ensure that no unintended side effects have been introduced.

If anomalies uncovered during testing can be repeated, but a decision has been made not to fix them, then these anomalies need to be evaluated in relation to the hazard analysis to verify that they do not affect safety of the device. The root cause and symptoms of the anomalies should be understood, and the rationale for not fixing them should be documented.

* 1. **Establish Tests for Software Requirements**

A set of tests, expressed as input stimuli, expected outcomes, pass/fail criteria and procedures, for conducting software system testing shall be established and performed such that all software requirements are covered.

* 1. **Use Software Problem Resolution Process**

Anomalies found during software system testing shall be entered into a software problem resolution process.

* 1. **Retest after Changes**

When changes are made during software system testing, the following shall be completed:

* Repeat tests, perform modified tests or perform additional tests, as appropriate, to verify the effectiveness of the change in correcting the problem
* Conduct testing appropriate to demonstrate that unintended side effects have not been introduced
* Perform relevant risk management activities
  1. **Verify Software System Testing**

The following shall be verified:

* The verification strategies and the test procedures used were appropriate
* Software system test procedures traced to software requirements
* All software requirements have been tested or otherwise verified
* Test results met the required pass/fail criteria
  1. **Software System Test Record Contents**

System test records shall contain:

* Documentation of the test result (pass/fail and a list of anomalies)
* Sufficient records to permit the test to be repeated
* The identity of the tester

1. **Software Release**

The objective of software release is to ensure all development, documentation, and testing is completed prior to release of the software.

* 1. **Ensure Software Verification is Complete**

The software verification shall be completed and the results evaluated before the software is released.

* 1. **Document and Evaluate known Residual Anomalies**

All known residual anomalies shall be documented and evaluated to ensure that they do not contribute to an unacceptable risk.

* 1. **Document Released Versions**

The version of the software product that is being released shall be documented.

* 1. **Document How Released Software was Created**

The procedure and environment used to create the release software shall be documented.

* 1. **Ensure Activities and Tasks are Complete**

All activities and tasks shall be ensured to be completed along with all the associated documentation.

* 1. **Archive Software**

The following shall be archived for at least a period of time determined as the longer of; the life time of the device or a time specified by relevant regulatory requirements:

* The software product and configuration items
* The associated documentation
  1. **Assure Repeatability of Software Release**

Procedures to ensure that the released software product can be reliably delivered to the point of use without corruption or unauthorized change shall be established. These procedures shall address the production and handling of media containing the software product including as appropriate:

* Replication
* Media labeling
* Packaging
* Protection
* Storage
* Delivery

1. **Software Configuration Management**

The following process is utilized to identify and control software versions and configuration items from development through implementation and obsolete software archiving. All historical versions of controlled configuration items and system configurations shall be maintained and retrievable.

* 1. **Software Configuration Identification**
     1. **Identification of Configuration Items**

All software configuration items shall have a unique identification. Product firmware versioning employs a number with two decimal points that is incremented by 0.01 for each change implemented (i.e. v1.36).

* + 1. **SOUP Identification**

The following data shall be documented for each SOUP configuration item being used, including standard libraries:

* + - * The Title
      * The Manufacturer
      * The Unique SOUP Designator
    1. **System Configuration Documentation Identification**

The set of configuration items and their versions that comprise the software system configuration shall be documented.

* 1. **Software Change Control**

All software changes are managed through the Change Control Process (QP-0009) and documented on the Engineering Change Order (ECO) form.

1. **Revision History**

| **Rev #** | **Doc #** | **Effective Date** | **CHO** | **Description of Change** |
| --- | --- | --- | --- | --- |
| 01 | QP-0010 |  |  | Initial release of the software development process. |