Quality System Procedure

Design and Development Procedure

PHILIPS

1 Purpose

The purpose of the Design and Development procedure is to describe and document all activities required for defining and developing the software only product, and related deliverables in terms that allow an adequate evaluation of conformance of design output to design inputs.

2 Scope

This procedure is applicable to the development of new products as well as implementation of modifications or improvements to existing products.

The procedure describes details of the “Develop Software Product incrementally” activities as defined in the PDLM procedure [0330-01-P].

The procedure does not cover activities, such as:

- Eliciting and prioritizing request for change. See Design Change Control procedure [0339-02-P].
- Research, advanced development and other innovation activities which are meant to demonstrate feasibility for new technologies, products and/or product enhancements.

3 Responsibilities

QMS Role	Responsibilities

QMS Role	Responsibilities
Product Manager	• Creates the User Requirements Specification • Defines and prioritizes the PROGRAM BACKLOG consisting of features to be included in the product along with Product management team • Reviews the Product Requirements Specification
Product Owner	Defines the TEAM BACKLOG by refining assigned features to prioritized user stories, in collaboration with Product Manager and Software Architect.
Requirements Analyst	• Responsible for creating the Product Requirements Specification • Ensures that the requirements from User Requirements Specification are traceable to Product Requirements Specification • . • Ensures that the risk mitigations indicated in the Risk Management File are mapped to the requirements contained in the PRS
Software Architect / Usability Architect	• Responsible for defining the product software architecture and design, which is documented in the System Design Specification. • Responsible to perform safety and security risk assessment for the product
Software Engineer	• Implements software • Defines Unit test • Defines Integration test • Executes Unit and Integration tests and documents results, including submission of engineering anomalies / anomalies, where applicable
Test Engineer	• Defines Verification test Specification • Executes verification tests and documents results, including submission of anomalies where applicable
Validation Manager	Responbile for execution of design validation activities as per Design Validation Procedure [0313-01-P].
Verification Manager	Responbile for execution of design verification activities as per Design Verification Procedure [0312-01-P].
Scrum Master	• Facilitates / supports PRODUCT DEVELOPMENT TEAM(s) during planning, execution and retrospective review of their activities • Supports Release Train Engineer during increment planning event and monitoring of progress
Release Train Engineer	• Facilitates increment planning event • Creates Development Increment Plan • Creates Development Increment Report • Monitors progress of planned activities during the increment across PRODUCT DEVELOPMENT TEAM(s)
Release Manager / Program manager / Project Manager	• Plans and manages end-to-end (cross-functional) product release activities (not part of this procedure) • Collaborates with Release Train Engineer and Product Manager in order to align the product release activities and priorities with product development
Technical Writer	• Responsible for establishing and maintaining the user documentation
Product Support Engineer	• Responsible for establishing and maintaining the Service documentation

4 Definitions and Acronyms

4.1 Definitions

Term	Definition
Design Change	A change to the DHF / DMR of a product. This can be a change to the product due to field issues and /or change due to requested enhancements or new functionality.
Engineering Anomaly	Anomaly in the product and its functionality that is still under development. These are typically detected prior to Start of Final Verification execution.
Product Development Team	Multi-disciplinary team typically including Requirements Analyst, Scrum Master, Product Owner, Software Architect, Software Engineers, Test Engineers, Technical Writer and Product Support Engineer.
Program Backlog	A repository for all upcoming work for a product. The PROGRAM BACKLOG consists primarily of future features intended to address user needs and accepted DESIGN CHANGES, but also includes architectural and/or service features and any other requirements to address needs of a stakeholder.
SOUP	Software Of Unknown Provenance (SOUP) is a software item that is already developed and generally available and that has not been developed for being incorporated into the medical device (also known as “off-the-shelf software”) or software item previously developed for which adequate records of the development processes are not available.
Team Backlog	The team backlog represents the collection of all work assigned to the team based on prioritized PROGRAM BACKLOG. The team backlog consists primarily of user stories and tasks required to implement features as defined in the PROGRAM BACKLOG.
FMEA	Failure Mode and Effects Analysis (FMEA) is a step-by-step approach for identifying all possible failures in a design, a process, or a product or service.

4.2 Acronyms

Short	Long
DHF	Design History File
DMR	Device Master Record
FMEA	Failure Mode Effects Analysis
PDLM	Product Development Launch & Maintenance
PMT	Program Management Team
PRC	Product Release Committed
PRS	Product Requirements Specification
RfC	Request for Change
RI	Release Initiation
SDS	System Design Specification
SFV&V	Start Final Verification and Validation
SOUP	Software Of Unknown Provenance
URS	User Requirements Specification

5 Procedure

5.1 Evolutionary/Incremental software product development

The development of software products follows the evolutionary/incremental development lifecycle model meaning that product is developed in increments while allowing definition to evolve over time, with the product continually refined and rebuilt. Evolutionary development model acknowledges that we do not understand all the requirements and build only those that are well understood. In this model, product requirements are partially defined upfront and then refined in each succeeding increment. The product and related deliverables are defined, built and verified in small pieces using a synchronized cadence of increments, managed via prioritized backlog, increment planning and acceptance criteria with the goal to deliver highest value for customers/users.

Following the principles of evolutionary/incremental development lifecycle model for medical device software means that:

- Development of design output (product and all required deliverables) can start before all design input requirements are finalized. However, design inputs must be finished/approved before design outputs can be finished/approved (following finish-finish relationship rather that finish-start relationship).
- Verification of developed functionality can be executed during increments if design inputs for that functionality are approved, and by gathering objective evidence through increments to demonstrate proper level of design controls.
- Impact of (later) changes to existing deliverables must be assessed and managed effectively. This requires regular synchronization and baselining of design input and design output (at increment and at release boundaries) as well as regression testing.

5.2 Process Flow

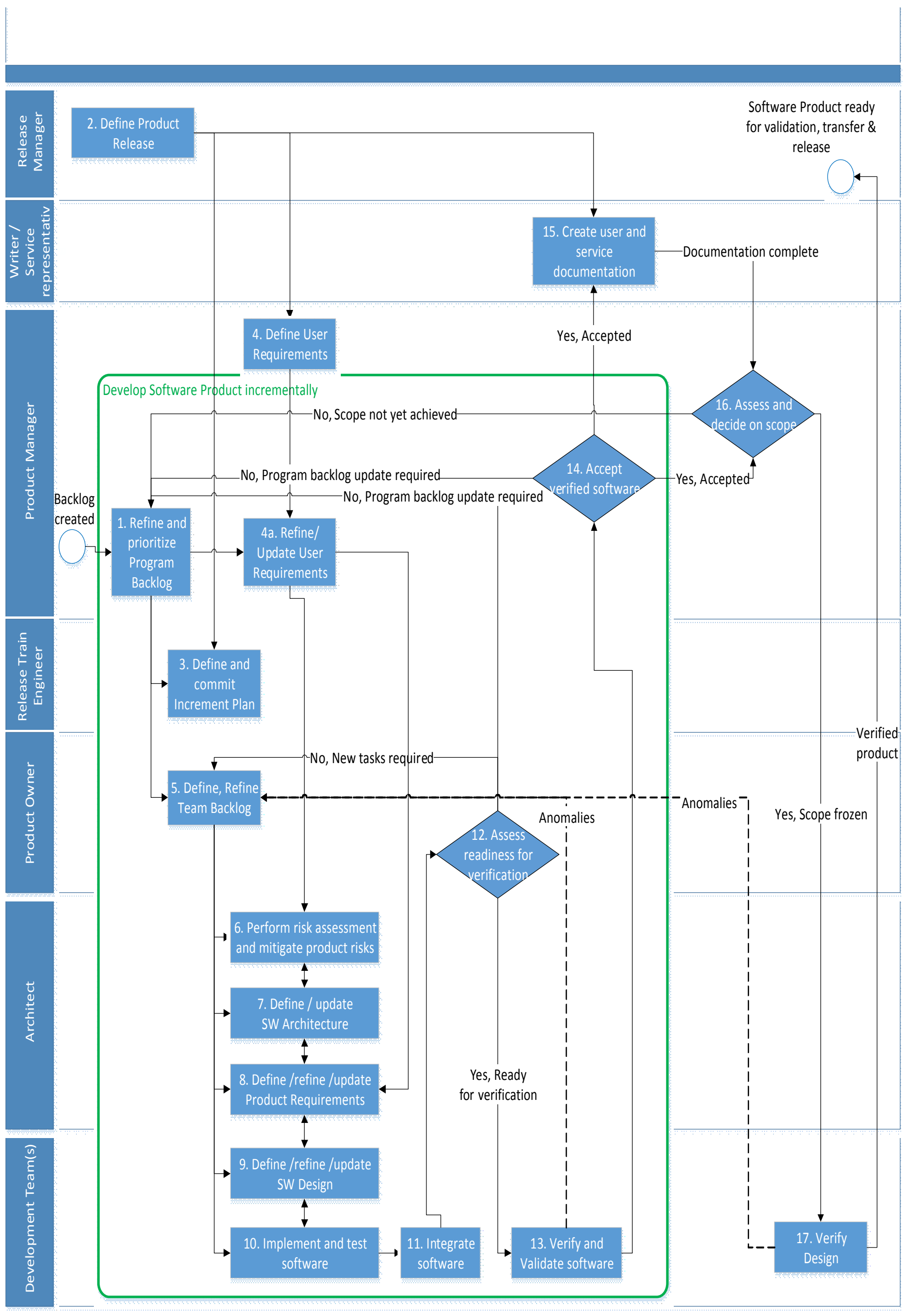
Release Manager
Writer / Service representativ
Product Manager
Release Train Engineer
Product Owner
Architect
Development Team(s)
2. Define Product Release
Software Product ready for validation, transfer & release
15. Create user and service documentation
Documentation complete
Yes, Accepted
4. Define User Requirements
Develop Software Product incrementally
No, Scope not yet achieved
16. Assess and decide on scope
14. Accept verified software
No, Program backlog update required
No, Program backlog update required
Yes, Accepted
Backlog created
1. Refine and prioritize Program Backlog
4a. Refine/ Update User Requirements
3. Define and commit Increment Plan
No, New tasks required
Verified product
5. Define, Refine Team Backlog
Anomalies
Anomalies
Yes, Scope frozen
12. Assess readiness for verification
6. Perform risk assessment and mitigate product risks
7. Define / update SW Architecture
8. Define /refine /update Product Requirements
Yes, Ready for verification
9. Define /refine /update SW Design
10. Implement and test software
11. Integrate software
13. Verify and Validate software
17. Verify Design

Figure 2: Flowchart diagram depicting the inter-relationship between “Develop Software Product Incrementally”, “Define Product Release” and “Verify Designs” blocks in the PDLM procedure [0330-01-P]. Activities within the green frame cover “Develop Software Product Incrementally” as per PDLM procedure [0330-01-P] and will be repeated until release scope is reached and final verification can start.

5.3 Pre-conditions

The Design and Development activities described in this procedure can start when:

- The initial PROGRAM BACKLOG is created and available (with identified and prioritized product needs)
- Organization including all roles, their capacity and competencies required for development of product is place, individuals for different roles are assigned and properly trained.
- Required development infrastructure is in place as per Lab Management procedure [0352-03-P] and Product Development Infrastructure procedure [0352-04-P].

5.4 Process Description

The details of each step of this workflow is explained in the table below:

Process Steps	Description	Deliverable
1. Refine and prioritize PROGRAM BACKLOG	**Responsibility :** Product Manager/Software Architect The Product Manager, supported by a multidisciplinary team representing all relevant stakeholders, regularly reviews, analyses features that can be implemented and demonstrated in an increment. The features are documented in the PROGRAM BACKLOG. If applicable, the Software Architect extracts enablers (architectural features) that are necessary to support upcoming features and/or address architectural risks, and prioritizes those in the backlog for implementation along features. The architectural features are documented in the PROGRAM BACKLOG. If applicable, the Product Manager, supported by a multidisciplinary team, executes an impact assessment for requested DESIGN CHANGES on existing software product and deliverables (e.g. requirements, design, code, safety/ security/privacy risk management file, verification and validation results, usability and regulatory aspects). Based on the impact assessment, a change request may be inserted in the PROGRAM BACKLOG to fix issues and/or new features may be defined to address DESIGN CHANGE. The Product Manager refines and prioritizes the features and/or enablers in the PROGRAM BACKLOG, in line with product release priorities.	NA
2. Define Product Release	**Responsibility:** Release Manager / Program manager / Project Manager The Release Manager provides the release specific objectives and priorities to the PRODUCT DEVELOPMENT TEAM(s) as part of the "Define Product Release". See Design and Development Planning Procedure [0334-01-P]. This input is used during the increment planning to define Development Increment Plan [0334-01-T8] to ensure that content and priorities of the product release are properly addressed and capacity required for release activities is allocated. The Release Manager ensures that the Development Increment Plan [0334-01-T8] is aligned to the existing Product Release Plan [0334-01-T1] at PRC milestone and at each increment boundary.	Product Release Plan [0334-01-T1]

Process Steps	Description	Deliverable
3. Define and commit Increment Plan	**Responsibility:** Release Train Engineer The Release Train Engineer, along with Scrum Masters, defines the Development Increment Plan for a given increment, based on prioritized PROGRAM BACKLOG, indicating features committed during the increment planning event. The selected scope is based on the capacity of the PRODUCT DEVELOPMENT TEAM(s), while taking into account factors such as capacity needed for architectural work, exploration/feasibility activities, support or product release related activities, while taking into account results of the previous increments (if applicable). The Development Increment Plan [0334-01-T8] shall be defined and approved at the start of each increment and in accordance with the Design and Development Planning Procedure [0334-01-P].	Development Increment Plan [0334-01-T8]
4. Define User Requirements	**Responsibility:** Product Manager The Product Manager defines or updates user requirements including acceptance criteria, to address the current release scope. The definition and update includes assessing the impact of new requirements on existing requirements and addressing any conflicts related to changes to existing requirements. The Product Manager defines User Requirements including acceptance criteria, in accordance with the Design Input Procedure [0332-01-P]. The User requirements are input for the product requirements definition and the product (safety and security) risk assessment as per Product Life cycle Risk Management Procedure [0336-01-P] and Product Security Risk Management Procedure [0336-02-P].	User Requirements Specification (URS) [0332-01-T1]
4a. Refine / Update User Requirements	**Responsibility :** Product Manager The assessment of the Product Backlog may lead to new or to adjustment of existing User Requirements, refinement / adjustment needs to address any conflicts related to existing requirements. The Product Manager refines / updates the User Requirements in accordance with the Design Input Procedure [0332-01-P].	User Requirements Specification (URS) [0332-01-T1]
5. Define / refine TEAM BACKLOG	**Responsibility:** Product Owner The Product Owner, in collaboration with the relevant PRODUCT DEVELOPMENT TEAM(s) continuously refines features/enablers from PROGRAM BACKLOG. The team elaborates the features / enablers into user stories and tasks, which together achieve the acceptance criteria of the	NA

Process Steps	Description	Deliverable
	required level of done for the features / enablers. Defined tasks are documented in the TEAM BACKLOG and need to address activities regarding requirements definition, safety/ security/ privacy risk management, architecture and design definition, implementation, testing, integration as well as verification.	
6. Perform risk assessment and mitigate product risks	**Responsibility:** Software Architect / Software Engineer The Software Architect, in collaboration with relevant stakeholders performs a safety risk assessment on the requirements and design in order to identify safety risks and to define risk control measures (additional product requirements and design constrains) for product to mitigate these risks. Product safety risk assessment and control is executed according to Product Life cycle Risk Management Procedure [0336-01-P]. The Software Architect in collaboration with relevant stakeholders performs security/ privacy risk assessment on the requirements and design in order to identify potential risks and to define any additional requirements and design constraints for the software to mitigate these risks. Product security risk assessment and control is executed according to Product Security Risk Management Procedure [0336-02-P].	Deliverables as per Product Life cycle Risk Management Procedure [0336-01-P] and Product Security Risk Management Procedure [0336-02-P].
7. Define/ update SW architecture	**Responsibility:** Software Architect The Software Architect defines / updates software architecture to address architectural needs to enhance design, performance, reliability and usability. Software Architecture may include definition of software structure and decomposition/sub-division of software into software items (components and units) necessary for risk control, interfaces between software items, User Interaction design, in-product 3rd party software, including identification of SOUP, interfaces to external systems and libraries. **SOUP Management** The software architecture shall cover or reference following aspects of SOUP: a) SOUP items used in the product shall be identified in the Third party assessment checklist [0334-01-T9]. b) Classification of the SOUP items (class A, B & C) to ensure that the risks associated with SOUP are managed and mitigated adequately c) For SOUP items classified as class B & C, specify the: - Functional & performance requirements of SOUP item - System hardware & software required to support	System Design Specification [0309-01-T1], Third party assessment checklist [0334-01-T9], Design Review Record [0315-01-T1], Deliverables as per Product Life cycle Risk Management Procedure [0336-01-P]

Process Steps	Description	Deliverable
	SOUP item d) For SOUP items classified as class B & C, capture the integration of SOUP with the product e) For SOUP items classified as class B & C, perform SOUP anomaly review: - SOUP anomalies associated with version used in the medical device should be reviewed to determine if any of the known anomalies result in a sequence of events that could result in the hazardous situation. - SOUP items used in the software product must be evaluated on a periodic basis to assess if the SOUP manufacturer or vendor has identified anomalies that could contribute to a hazardous situation or affect the effectiveness of the risk mitigation. - The strategy for the periodic review and its frequency should be clearly identified to ensure continuity of the SOUP item usage and determine further course of required (if any). - The objective evidence for the SOUP anomaly review should be managed as part of the Product DHF. Refer to the Annex A: Guidance on SOUP Management for additional details. **Static Verification of Architecture:** Static verification of software architecture should be performed to verify that a) The software architectural design implements product requirements including those relating to risk control measures. b) The software architectural design supports interfaces between both software items and between software items and hardware. c) The software architectural design supports proper operation of SOUP items. Objective evidence of static verification of architecture is to be demonstrated through Design Review Record [0315-01-T1] as per Design Review Procedure [0315-01-P].	
8. Define/ refine/ update Product Requirements	**Responsibility:** Requirements Analyst The Requirements Analyst defines, refines, and/or updates the Product Requirements, including acceptance criteria for the features under development. The impact of new requirements on existing requirements must be assessed and addressed including traceability. The requirements includes functional requirements as well as quality aspects requirements. The Requirements Analyst shall define the Product Requirements in accordance with the Design Input Procedure [0332-01-P].	Product Requirements [0332-01-T2],

Process Steps	Description	Deliverable
	Inputs for creating the Product Requirements are: • User Requirements • Product security risks and identified risk controls measures • Product safety risks including usability related safety risks and their identified risk control measures • Program / TEAM BACKLOG • Existing product requirements Ensuring adequacy of the Product Requirements (in sufficient detail) is part of the refining activities and must be completed before starting verification of requirements. The requirements traceability shall be updated and documented, along with product requirements as per Design Input Procedure [0332-01-P].	
9. Define / refine / update SW Design	Responsibility: Software Architect, Software Engineer, Usability Architect The Software Architect / Software Engineer continuously defines, refines and/or updates Software Design, in accordance with the requirements, User Interface and architecture. User Interface specification shall consider • Use Specification captured in the URS, • known or foreseeable Use Errors associated with the product • and Hazard related Use Scenarios; User Interface specification will include: • User Interface requirements, including those which are related to the Risk mitigations • associated reference to the Accompanying Documentation (IFU/User manual) and , • an indication whether product specific training is required. Updating of software design provides information for execution of product safety risk assessment and product privacy security risk assessment, input for assessment and updating the User Interaction design, algorithm identification (if applicable). The software design shall be kept consistent with the product requirements. Failure Mode and Effects Analysis (FMEA) shall be considered for critical functionality during design to identify possible risk of design failures, and to identify the required actions to address such failure where necessary. The impact of design failure and identification of required actions on the broader system components (which may include Service Tools, plug-ins, 3rd party components, hardware etc.) should be part of this consideration.	(Updated) SDS [0309-01-T1], UI Specification [0730-02-T1] as per Usability Engineering Guidance [0331-01-G1] Design review record [0315-01-T1]

Process Steps	Description	Deliverable
	For software classified as Class C only: Software design is documented in the System Design Specification along with Software Architecture. SDS shall capture sufficient details of design with enough detail to allow correct implementation of each software unit and its interfaces. **Static Verification of Detail Design (For Class C only):** Static verification of software detail design should be performed to verify that - The software detailed design implements the software architecture - The software detailed design is free from contradiction with the software architecture Objective evidence of static verification is to be demonstrated through Design review record [0315-01-T1] as per Design review procedure [0315-01-P]. The (updated) SDS is reviewed and approved in accordance with the General Document Control Procedure [0730-02-P].	
10. Implement and test software	**Responsibility:** Software Engineer The Software Engineer implements the software code in accordance with the defined product requirements and software design. The Software Engineer ensures code review is executed by peer Software Engineer and ensures that code review results are recorded. See Code Review Guidance [0309-01-G1]. The code review record includes: • Commit revision • Commit Log • Review Id • Review title • Identification of reviewer(s) and date of review • Review status **Software Unit Verification** • Documentation of unit test shall be done for software units classified as Class B and C and the objective evidence of the unit verification implemented shall be maintained as part of product DHF. • Establish strategies, methods and procedures for verifying each software unit, as appropriate. • The methods for the unit verifications may include code reviews, static code analysis, Dynamic code analysis, Test automation, etc. • For class B and C, Software unit acceptance criteria shall be met prior to integration into larger software items as appropriate to ensure product safety. Refer to Annex B:	Implemented SW Code Review records Unit test records (if applicable)

Process Steps	Description	Deliverable
	Software Unit Acceptance Criteria for the list of the acceptance criteria. • Approve Unit test specification prior to execution for all new application or features and their modification. Approved Unit test cases can be used for multiple releases unless and until there is change needed to them because of change in requirements. • The Software Engineer executes the unit test and solves engineering anomalies found during the testing, before delivering the reviewed and tested software to the software archive. • Engineering anomalies uncovered during unit tests are to be handled as per section 5.5 of this document. • Unit test specification and test report are reviewed and authorised according to the General Document Control procedure [0730-02-P].	
11. Integrate software	**Responsibility:** Software Engineer / Test engineer The Software Engineer integrates implemented and tested software units into software items/systems. Integration of software is performed continuously during software development. The Software Engineer solves any engineering anomalies found during integration before starting verification activities. In case of complex systems and/or broader system components (which may include the 3rd party components, hardware etc.) the additional integration and testing of integrated software is done at pre-defined increments and/or during the final product verification. Documentation of Integration test shall be available as per Integration plan (may be part of the Development Increment Plan [0334-01-T8]) for software Units classified as Class B or C as per IEC 62304 Software Risk Classification. It is acceptable to combine integration testing and software verification into a single plan and set of activities. If Integration testing is combined with system testing, then see Design Verification Procedure [0312-01-P] for roles involved, applicable activities and related deliverables. The Software Engineer or Test Engineer defines integration test cases and ensures these are reviewed and approved for adequacy before execution of test. The Software Engineer or Test Engineer performs integration testing of the integrated software and documents the results including any anomalies uncovered. These anomalies shall be handled as per Defect Management Procedure [0339-01-P]. The test specification and test report for Integration (can be part of the Verification test specification and Report) are reviewed and authorised according to the General Document Control procedure [0730-02-P].	Integrated SW Integration test records (if applicable)

Process Steps	Description	Deliverable
12. Access readiness for verification	**Responsibility:** Product Owner The Product Owner assesses if the team has achieved the required level of done (based on pre-defined “definition of done”) required to start verification for implemented and tested software. “Definition of done” is a team checklist indicating all relevant aspects of the development (including implementation, documentation update, integration and testing) to indicate if development is completed and verification can start. This includes: • Completed requirements for software to be verified • Completed software design for software to be verified • Implemented and baselined software • Completed safety, privacy and security risk management matrix for software to be verified based on the assessment of the completed requirements and design • Completed implementation of safety, privacy and security risk mitigations for software product to be verified • Completed verification test cases for the software product to be verified • Completed associated documentation with the above activities	NA
13. Verify and Validate software	**Responsibility:** Test Engineer / Validation manager The Test Engineer verifies software against the defined and approved product requirements and according to Development Increment Plan. The Test Engineer defines verification test cases, including identification test steps and expected results, specification of acceptance criteria and required test environment in relation to the product requirements. Verification test cases shall be traceable to the product requirements. Test Engineer executes planned verification test cases, analyses and documents the results of verification activities, including anomalies found during the verification. Verification of software includes verification of interaction and interfaces of already developed software (functional tests as well as some non-functional tests). To demonstrate that changes from the current software do not affect software verified in previous increments, regression testing along with verification of new functionality is part of this activity. Verification shall be executed and objective evidence of verification shall be gathered and recorded as per Design Verification Procedure [0312-01-P]. Any anomalies uncovered during product verification shall be handled as per Defect Management Procedure [0339-01-P].	Approved Verification Test Cases [0312-01-T2] or [0312-01-T5] Verification test records [0312-01-T3] or [0312-01-T6] Approved Validation protocol [0313-01-T1] Validation report [0313-01-T2]

Process Steps	Description	Deliverable
	The planned validation activities, if any, shall be performed on the verified software in accordance with the Design Validation procedure [0313-01-P]. Clinical evaluation may be performed for the features under development, including algorithms (if applicable) as per Clinical Evaluation Procedure [0301-01-P].	
14. Accept verified software	**Responsibility:** Product Owner The Product Owner, along with Product Manager decides if integrated and verified software can be accepted, based on the defined acceptance criteria in the definition of done. The software is not accepted if the verification results indicate that there are open anomalies, or implementation does not address the defined requirements properly, requirements are not adequate etc. If such case the program/TEAM BACKLOG may be updated to indicate required additional work. The results of the Development Increment are recorded in the Development Increment Report [0334-01-T8]. The Development Increment Report is reviewed and authorised according to the General Document Control procedure [0730-02-P].	Development Increment Report [0334-01-T8]
15. Create user and service documentation	**Responsibility:** Technical Writer / Service representative The Product Requirements, User Interaction Design and its implementation are input for the creation of user documentation. User documentation shall be created by the Technical writer, reviewed and approved according to Creation and Localization of User Documentation procedure [0319-01-P]. The Product requirements, system design and its implementation are input for the creation of service documentation. Service documentation shall be created, reviewed and approved according to Creation of Service Documentation procedure, see [0318-02-P]. • Service documentation includes: Service manuals to be used by the service organization, including product installation / upgrade instructions and any requirements for media carriers and build/pre-installation tooling, product service instructions and product's performance assurance instructions • Training material for the service organization	User documentation as per Creation and Localization of User Documentation procedure [0319-01-P]. Service documentation as per Creation of Service Documentation procedure [0318-02-P]
16. Assess and decide on scope	**Responsibility:** Product Manager (content) & Release Manager / Program manager / Project Manager (deliverables) At SFV&V milestone, the Product Manager, in interaction with the Release Manager and PRODUCT DEVELOPMENT TEAM(s) assesses	Milestone Review Report [0334-02-T1] for

Process Steps	Description	Deliverable
	implemented functionality and decides if the scope required for the product release is achieved and freeze the release scope (requirements). The related design inputs and design outputs, including user documentation and service documentation are finished so final verification and validation can start.	SFV&V milestone as per [0334-02-P]
17. Verify Designs	**Responsibility:** Verification manager Final Design Verification activities must demonstrate that the final product satisfies its requirements. The verification done in earlier stages during increments may be considered and accepted when defining content of the final verification activities. This is possible when: Verification of software increment has been performed as per Design Verification Procedure [0312-01-P]. Changes to the software during subsequent increments did not impact already executed verification activities and approved deliverables (requirements, implementation or verification test cases / results). If this is not the case, verification activities shall be repeated during the final verification. The verification manager assesses already executed verification during incremental development and indicates and plans any additional verification activities (including required re-execution of verification tests) to be executed during final verification. The Verification manager prepares the start of final verification by assessing verification readiness as per Design Verification Procedure [0312-01-P]. Verification Team executes final verification of the product in line with Verification Plan to demonstrate fulfillment of product requirements including risk mitigations. Design Verification activities shall be performed in accordance with the Design Verification Procedure [0312-01-P]. Any anomalies uncovered during product verification (of class A,B,C) shall be handled as per Defect Management Procedure [0339-01-P]. After the verification activities are finalized and results evaluated, the product is ready for validation as per Design Validation procedure [0313-01-P], transfer as per Design Transfer procedure [0317-01-P] and release as per Release for Delivery procedure [0314-01-P].	As per Design Verification Procedure [0312-01-P]

5.5 Handling Engineering anomalies

Engineering anomalies obtained prior to the start of design verification activities such as at planning, requirements, design, Implementation, etc., are to be managed adequately for its resolution and closure.

Engineering anomalies related to functionality under development and/or uncovered during unit tests can be managed through TEAM BACKLOG. All engineering anomalies kept open at the start of final verification and validation activies, shall be considered as a product defect and will be managed through the requirements of Defect Management Procedure [0339-01-P].

Engineering anomalies uncovered pertaining to features that are already existing in released product / functionality shall be handled as per Defect management procedure [0339-01-P] and assessed to determine what action is required in case potential safety defect is part of a released product.

6 Records

Record Name	Description
System Design Specification [0309-01-T1]	Template to capture the System Design Specification of the product
FMEA Template [0309-01-T4] (Optional)	Template to capture the Failure Mode and Effect Analysis for impact assessment of SW DESIGN CHANGES

7 References

Reference Number	Description
0309-01-G1	Code Review Guidance
0312-01-P	Design Verification Procedure
0312-01-T2 or 0312-01-T5	Verification Test specification
0312-01-T3 or 0312-01-T6	Verification Test Results
0313-01-P	Design Validation procedure
0313-01-T1	Validation Protocol
0313-01-T2	Validation Report
0314-01-P	Release for Delivery Procedure
0315-01-P	Design Review Procedure
0315-01-T1	Design Review Record
0317-01-P	Design Transfer Procedure
0318-02-P	Creation of Service Documentation procedure
0319-01-P	Creation and Localization of User Documentation procedure
0330-01-P	Product Development Launch & Maintenance (PDLM) Procedure
0330-01-G1	Usability Engineering Guidance
0332-01-P	Design Input procedure
0332-01-T1	User Requirements Specification
0332-01-T2	Product Requirements Specification
0334-01-P	Design and Development Planning Procedure
0334-01-T1	Product Release Plan
0334-01-T8	Development Increment Plan and Report

Reference Number	Description
0334-01-T9	Third party assessment checklist
0334-02-P	Milestone and Gate review procedure
0334-02-P	Milestone Review procedure
0334-02-T1	Milestone Review Report
0336-01-P	Product Life Cycle Risk Management
0336-02-P	Product Security Risk Management Procedure
0339-01-P	Defect Management Procedure
0340-01-P	Design Output Procedure
0352-03-P	Lab Management procedure
0352-04-P	Product Development Infrastructure procedure
0730-02-P	General Document Control Procedure

8 Document History

Revision	Description of changes
1.0	Initial Release
2.0	• Service Innovation Engineer changed to Product Support Engineer • Scrum Master and Product Owner added to the PRODUCT DEVELOPMENT TEAM • More explicit indicated to assess the impact of requirements on already existing product (design, safety and security risks and mitigation, verification and validation) • Product backlog added as input for this process • Iteration Plan and Report template added to Annexes
3.0	Added reference to code review guidance document
4.0	Updated iteration plan content in § 4.2
5.0	Editorial changes (references updated)
6.0	Administrative change to correct version number.
7.0	Procedure updated to harmonize with HISS-W-030002.12 (Ref C) work instructions. Updated template to align with Rev 2.0 of the 0730-01-T1.
8.0	Administrative change: Removed DRS/DDS references as it is not applicable to the organization. In absence of DDS, clarified that detailed design is captured in SDS document and is necessary only for software with Class C as per IEC 62304. Note: The SDS template [0309-01-T1] Rev 2.0 already captures this detail and the procedure is updated with this missing information.
9.0	Section 5.4, step 9: Inclusion of FMEA methodology as an optional method for impact assessment during SW design implementation.

9 Document Control

Process	**Owner(s)**
Approval	Q&R manager Process Owner PCP
Review	Development Manager
Author	Moshe Elmaliach
Approval date	See eDMS
Effective date plan	According to the following implementation plans: EII.0004734 and EICI.0007439

10 Annexes / Appendices

10.1 Annex A: Guidance on SOUP Management

10.1.1 Classification of SOUP items

Each SOUP is assigned a safety classification.

If SOUP contributes to potential cause of a hazardous situation, the SOUP would be
- CLASS C for a hazard with severity critical or catastrophic
- CLASS B for hazard with severity marginal or negligible.

If SOUP does not contribute to potential cause a hazardous situation, it is CLASS A.

10.1.2 Functional and Performance requirement of SOUP

Each SOUP item has functional and performance requirements, similar to other items in the architecture. These requirements may be specified when necessary for the SOUP item to fulfill its intended use in the product and verification performed to ensure they are met.

Functional requirements specify the expected behavior of the SOUP for its intended use in the product. Often, these requirements will identify key features of the SOUP that can be verified to ensure that the SOUP performs its intended use.

Performance requirements specify criteria, which is essential for the safe and effective operation of the device.

Functional Requirements for SOUP items may include:
- Specifications for discrete-valued SOUP item inputs and outputs
- Specification of SOUP item behavior/Transfer functions

Performance Requirements for SOUP items may include:
- Specifications for numerical SOUP item inputs and outputs (e.g. range, accuracy and precision)
- Specification of SOUP item timing (e.g. latency, periodicity, etc.,) requirements related to executions of functions, servicing inputs, providing outputs, etc.,
- Specification of its SOUP item resource utilization (memory footprint, etc.,)
- Risk Mitigations related to the SOUP item

When requirements are identified for SOUP items, they should be clearly specified similar to requirements for other items in the system. Some methods for specifying SOUP requirements include: having a SOUP requirements specification that contains all of the SOUP requirements or tagging a system requirements as applicable to a SOUP item or writing a requirement for the SOUP as part of a software requirements. The strategy should be clearly identified and to be followed during the program execution.

10.1.3 Hardware and Software Necessary to support SOUP

Software architecture documents the product hardware and software configurations needed to support the operation of each SOUP. This can include the processor or chip version, memory, disk space, network bandwidth, operating system, middleware or other requirements of the SOUP. If open source or “internal” SOUP is used, there may also be requirements regarding compatibility of compilers or other build tools.

Hardware specifications may include, but not limited to:
- Processor type, capabilities, speed required to support SOUP items
- Memory type, speed and size required to support SOUP items
- Network connectivity (e.g. internet, WAN, LAN) required to support SOUP items
- Display and audio system capabilities required to support SOUP items

- Any other hardware that is required for the proper functioning of a SOUP items

Software specifications may include, but not limited to:
- Operating system type and configuration (Version, patches, etc.,) required to support the SOUP items
- Device drivers, services, Shim layers, wrapper APIs etc., required to support SOUP items
- Any other software items required for the proper functioning of a SOUP item

Typically, the versions and configurations are the minimum required for the operation of the SOUP. The information should be sufficient to ensure compatibility between the SOUP and the product. For example, a graphics library may be compatible with windows 7 but not with XP.

10.1.4 SOUP Anomaly Review

If failure or unexpected results from SOUP is a potential contributing cause to a hazardous situation, then the anomaly list published by the supplier of the SOUP item relevant to the version of the SOUP item is evaluated to determine if any of the known anomalies contribute to a sequence or combination of events that could result in a hazardous situation.

Periodic evaluation of SOUP assesses whether the SOUP manufacturer has become aware of any anomalies that could affect risk. The evaluation considers defects and changes to the SOUP to see if those changes or defects could:
- Introduce new causes or hazardous situations
- Introduce a new sequence of events for an existing cause
- Impact the effectiveness of risk mitigations
- Change or impact a functional or performance requirement of the SOUP
- Require different hardware or software support

The information for the SOUP anomalies can be found in
- Notifications from SOUP manufacturer
- Published release notes and bug database
- Supplier change notices, where applicable to the supplier
- User Forums
- Internet searches

Below are the information, but not limited to, that can be included in the SOUP anomaly review record.

Information to include	Description
Name	Name of the SOUP
Version	Version reviewed
Date	Date of review conducted
Summary	Summary of review, including description of the anomalies or changes that were found that could impact the product
Conclusion	Determination if any additional actions are required. If no issues are found then no further action is required. Note: If no information around the SOUP is obtained through the sources identified above, then the conclusion should clearly document the efforts performed by the team including references to the sources for concluding on the non-availability of soup information and to drive further actions.
Action Plan, if needed	DESIGN CHANGE, Anomaly, or other tracking mechanism that could be used to identify any further actions. Actions could include upgrading or replacing SOUP or implementing a DESIGN CHANGE to the product or risk management related to a new hazardous situation.

10.2 Annex B: Software Unit Acceptance Criteria

Software unit acceptance criteria includes (CLASS B & CLASS C)

- Does the software code implement requirements including RISK CONTROL measures?
- Is the software code free from contradiction with the interface design of the SOFTWARE UNIT?
- Does the software code conform to programming procedures or coding standards, if applicable?

Additional Software unit acceptance to be considered, as appropriately, when present in the software product design (CLASS C)

- 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 and
- Boundary conditions

End of Document

PHILIPS
sense and simplicity

This is a representation of an electronic record that was signed electronically in our Regulated System.

This page is the manifestation of the electronic signatures used in compliance with the organizations electronic signature policies and procedures.

UserName: Balamurugan Gopal (ing11491)
Title: Group Leader
Date: Tuesday, 20 March 2018, 06:07 PM Pacific Daylight Time
Meaning: This document has changed to Authorized status
===

UserName: Karel Strasters (nly16220)
Title: Senior Manager R&D
Date: Thursday, 22 March 2018, 09:08 AM Central Europe Standard Time
Meaning: This document has changed to Authorized status
===

UserName: Emilly Nurthen (310234991)
Title: Head of Quality and Regulatory(EI)
Date: Monday, 26 March 2018, 12:30 PM Pacific Daylight Time
Meaning: This document has changed to Authorized status
===

UserName: Trudy Kunnen (nly22709)
Title: Director Quality & Regulatory
Date: Tuesday, 27 March 2018, 08:04 PM Central Europe Daylight Time
Meaning: This document has changed to Authorized status
===