SW Design

**Change History**

| **Version** | **Date** | **Change Description / Reason** | **Author** |
| --- | --- | --- | --- |
| 1.0 | 2017-12-08 | Initial version based on Hella Process (State Avenue)  Change Process name  Add Process procedure figure (High Level)  Add software architecture example figure  Add evaluate alternative software architecture in Task 2  Add dynamic behavior description in Task 7  Add objective field in all tasks. | Hae-Min, Woo |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[1. Purpose Description 4](#_Toc491876678)

[1.1 Purpose 4](#_Toc491876679)

[1.2 Main description 4](#_Toc491876680)

[1.3 Result of Process 4](#_Toc491876681)

[2. Major Roles Acting in this Process 4](#_Toc491876682)

[3. Process 5](#_Toc491876683)

[3.1 Process Input – Output Definition 5](#_Toc491876684)

[3.2 Introducing Aspects 6](#_Toc491876685)

[3.3 Process Flow Chart 6](#_Toc491876686)

[3.3.1 Task 1: Analyze Factors of Influence Concerning the SW Architecture 8](#_Toc491876687)

[3.3.2 Task 2: Develop Software Architecture 9](#_Toc491876688)

[3.3.3 Task 3: Allocate SW Requirements 12](#_Toc491876689)

[3.3.4 Task 4: Define Interfaces and Dynamic Behavior 13](#_Toc491876690)

[3.3.5 Task 5: Create Safety Analysis Report 13](#_Toc491876691)

[3.3.6 Task 6: Analyze Dependent Failures 14](#_Toc491876692)

[3.3.7 Task 7: Develop Detailed Design and Refine Behavior 14](#_Toc491876693)

[3.3.8 Task 8: Classification of the SW Modules and SW Units 15](#_Toc491876694)

[3.3.9 Task 9: Develop Verification Criteria 16](#_Toc491876695)

[3.3.10 Task 10: Ensure Traceability and Consistency to SW Architecture 17](#_Toc491876696)

[3.3.11 Task 11: Ensure Traceability and Consistency to SW Requirements 17](#_Toc491876697)

[3.3.12 Task 12: Verify SW Architecture and Module & Unit Design 17](#_Toc491876698)

[3.3.13 Task 13: Create and communicate baseline 18](#_Toc491876699)

[4. Changes, References, Appendix, Terms 19](#_Toc491876700)

[4.1 References 19](#_Toc491876701)

[4.2 Template 19](#_Toc491876702)

[4.3 Abbreviation 19](#_Toc491876703)

# Purpose Description

## Purpose

The purpose of the this procedure is to define the structure of the software to build, its interfaces, the overall behavior of the software and the allocation of requirements onto software elements in order to enable the software developer to create the source code.

## Main description

The input of the system architecture design and the software requirements specification is used in order to structure the software in modules (and units) and interfaces. The overall behavior of the software is defined and the requirements of the software requirements specifications are allocated onto the structural elements.

## Result of Process

As a result of the successful implementation of this process:

* A software architecture is defined that identifies the components of the software and meets the defined software requirements;
* The software requirements are allocated to the elements of the software;
* Internal and external interfaces of each software module are defined;
* The dynamic behavior and resource consumption objectives of the software modules are defined;
* A detailed design is developed that describes software units that can be implemented and tested;
* Consistency and bilateral traceability are established between software requirements and software architecture;
* Consistency and bilateral traceability are established between software architecture and detailed software design.

# Major Roles Acting in this Process

|  |  |
| --- | --- |
| **Role** | **Contribution and Responsibilities** |
| Software Architect | * Create Software Architecture Specification * Estimate and determine critical hardware resources * Monitor consistency between software architecture, software design * Execution of determining module classes of software modules * Develop test and verification criteria for software modules (software components) |
| Safety Manager | * Creation of the Safety Analysis Report * Creation of the Dependent failures report * Support of the SW Architect and software Developer concerning safety relevant topics |
| Software Project Manager | * Review various work results * Hold kick-off and team meetings * Create Status Reports |
| Software Developer | * Create detailed design * Execute changes that affect the unit design * Review changes to software Architecture concerning his or her software modules * Assist in determining modules classes of software modules |
| Software Integration Tester | * Support develop test and verification criteria for software modules (software components) * Review software architecture specification |
| Software Analyst | * Provide software requirements specifications * Jointly responsible for specifications of module classes for software modules |
| System Architect | * Review software architecture specification |
| Software Quality Planner | * Review process * Decides on determining module classes of software modules * Co-determine the module classes of various modules |
| Software Module Tester | * Review software module and unit design |

# Process

## Process Input – Output Definition

The following process input is required:

|  |  |
| --- | --- |
| **Process input** | **From supplier** |
| Software Requirements Specification | Software Analyst |
| System Architecture Specification | System Architect |
| System Interface Specification | System Architect |

|  |  |
| --- | --- |
| **Process output** | **To customer** |
| Software Architecture Specification | Software Integration Tester  Software Developer |
| Safety Analysis Report | Safety Manager |
| Dependent failures Report | Safety Manager |
| Review Checklists | Software Quality Planer |
| Software Module and Unit Design | Software Developer  Software Module Tester |

## Introducing Aspects

None

## Process Flow Chart

Software design process (High Level) is as below,



Software design process (Detail) is as below,





### Task 1: Analyze Factors of Influence Concerning the SW Architecture

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Input | | Software Requirements Specification | | | | | |
| Output | | Software Requirements Specification | | | | | |
| D: |  | E: | SW Architect | S: | SW Analyst  System Architect | I: |  |
| D = Decision | | E = Execution | | S = Support | | I = Information | |

**Objective**

The goal of this task is to identify the influence of the functional and no- functional requirements (quality attributes) on the Software Architecture.

**Description**

There are several functional requirements and quality attributes, such as performance, maintainability, functional safety, reusability, portability, testability or scalability, which can influence the Software Architecture.

**Note**, that the quality attributes usually have different priorities in the context of the project. As a prerequisite for the following tasks, the Software Architect has to analyze these requirements and attributes focusing on the development of Software Architecture. The Software Analyst must provide this information, which is part of the software requirements specification and the software analysis. Output of this task are(derived) Software Architecture specific requirements. These requirements are again part of the Software Requirements Specification.

### Task 2: Develop Software Architecture

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Input | | System Requirements Specification  Software Requirements Specification | | | | | |
| Output | | Software Architecture Specification | | | | | |
| D: |  | E: | SW Architect | S: | SW Analyst  Safety Manager | I: | SW Project Manager |
| D = Decision | | E = Execution | | S = Support | | I = Information | |

**Objective**

The goal of the architectural design is to develop an overall software structure which satisfies the given functional and non-functional software requirements.

**Description**

The software architecture structures the software of the system under development. It defines the elements of the software, describes the properties of these elements and their interactions. There are different views on the software architecture – each focusing on a different concern. The Software Architect shall specify the software architecture in views of:

* Architectural style
* Software modules (Software-components) and their dependencies
* Execution structure
* Interrupt handling

In case of an ASIL classified device under construction the architecture additionally has to respect

* Mechanisms for error detection
* Mechanisms for error handling

Note that the software architecture is mainly driven by the functional and non-functional software requirements. Thus, the development of the software architecture and the mapping of the according requirements to the specified elements of the software (Task 4: ) are done by the Software Architect in parallel.

**Architectural Style**

An architectural style is a set of constraints on architecture. These constraints describe how to assemble a set of elements (software modules and units) and how these elements interact with each other. For this a layered architectural style is to be used. If a layered style is used, all layers like applications (incl. manager), drivers and handlers need to be illustrated for simple systems one layer may be enough.

Example



Additionally, alternative software architectures should be considers and evaluate.

If the software under development has an ASIL classification the software architecture is to be documented in a semi-formal notation (e.g. UML/SysML).

**Software Modules and their Dependencies**

The software architect structures the software into (software-)modules. This also includes the identification of software modules that are already available for reuse[[1]](#footnote-1). Modules encapsulate data and functionality and provide a clearly defined interface to their environment. There are different factors that influence the way the software is divided into modules (see Task 1):

* Software modules should have a maximum cohesion and a minimal coupling.
* Each software module should be testable in isolation.
* Software modules should be reusable.
* Software modules should be restricted in size
* Integration requirements of third party software must be regarded

For each software module, the Software Architect must add a specification which roughly describes the purpose of the module. Additionally, resource consumption objectives, like memory consumption (ROM and writeable memory), stack usage, CPU usage (timing and load), interrupt usage and timing constraints are to be specified. Note, that if no data is available, the Software Architect must roughly estimate the resource consumption objectives. Moreover, to prepare the software module and software integration test, the specification of verification criteria (see also Task 11) must be added.

The software architect must keep the module design consistent with the other views of the software architecture. This also includes a clearly defined mapping between the elements of the different views, e.g. by using identical names or more sophisticated mechanisms provided by the used tools (e.g., linking).

If software partitioning is used to implement freedom from interference the usage of shared resources is to be restricted and shall be avoided (e.g. usage of dedicated hardware features)

**Execution Structure**

The execution structure describes the division of the system into tasks. Moreover, for each task, the task model specifies:

* priority,
* interruptability (pre-emptive or non-pre-emptive), and
* messages and events that are
* received;
* transmitted;
* set;
* or deleted.

The usage of interrupts is to be minimized.

If a time slice model is used instead of an operating system or emulated by it the description must specify which application runs in what time slice and in which way the various applications exchange communication.

**Mechanisms for error detection and handling**

Based on the results of Task 6: the mechanisms for error detection shown in Figure 2 and error handling shown in Figure 3 have to be implemented.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Methods** | | **ASIL** | | | |
| **A** | **B** | **C** | **D** |
| 1a | Range checks of input and output data | ++ | ++ | ++ | ++ |
| 1b | Plausibility check a | + | + | + | ++ |
| 1c | Detection of data error b | + | + | + | + |
| 1d | External monitoring facility c | o | + | + | ++ |
| 1e | Control flow monitoring | o | + | ++ | ++ |
| 1f | Diverse SW design | o | o | + | ++ |
| 1. Plausibility checks can include using a reference model of the desired behavior, assertion checks, or comparing signals from different sources. 2. Types of methods that may be used to detect data errors include error detecting codes and multiple data storage. 3. An external monitoring facility can be for example an ASIC or another SW element performing a watchdog function. | | | | | |

Figure 2: Methods for error detection. (“++” means “highly recommended” , “+” means “recommended”, “o” means no recommendation for or against its usage)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Methods** | | **ASIL** | | | |
| **A** | **B** | **C** | **D** |
| 1a | Static recovery mechanism a | + | + | + | + |
| 1b | Graceful degradation b | + | + | ++ | ++ |
| 1c | Independent parallel redundancy c | o | o | + | ++ |
| 1d | Correcting codes for data | + | + | + | + |
| 1. Static recovery mechanisms can include the use of recovery blocks, backward recovery, forward recovery and recovery through repetition. 2. Graceful degradation at the SW level refers to prioritizing functions to minimize the adverse effects of potential failures on functional safety. 3. Independent parallel redundancy can be realized as dissimilar SW in each parallel path. | | | | | |

Figure 3: Methods for error handling. (“++” means “highly recommended” , “+” means “recommended”, “o” means no recommendation for or against its usage)

### Task 3: Allocate SW Requirements

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Input | | Software Requirements Specification  Software Architecture Specification | | | | | |
| Output | | Traceability Record | | | | | |
| D: |  | E: | SW Architect | S: | SW Analyst | I: |  |
| D = Decision | | E = Execution | | S = Support | | I = Information | |

**Objective**

The objective is to allocate the software architecture to software requirements.

**Description**

The Software Architect must allocate all software requirements to the elements (software modules, tasks) of the software architecture. This means that particular elements of the software architecture including their detailed design and implementation will have to satisfy the software requirements allocated to them. Thus, the software requirements are mapped to the structure of the software architecture. Usually the tasks of developing the software architecture and of allocating the software requirements are performed in an iterative way.

**Note**, that in Task 12: bilateral traceability is required to support the verification of the software design. Thus, links between the software architecture (e.g., software modules in the design) and the software requirements must be traceable in both directions. This shows what requirements are satisfied by which elements of the software architecture and allows the Software Architect to determine if all software requirements were considered. Moreover, changing software requirements and the resulting impact on the software architecture will be immediately recognizable.

### Task 4: Define Interfaces and Dynamic Behavior

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Input | | System Interface Specification  Software Requirements Specification | | | | | |
| Output | | Software Architecture Specification | | | | | |
| D: |  | E: | SW Architect | S: |  | I: |  |
| D = Decision | | E = Execution | | S = Support | | I = Information | |

**Objective**

The objective is to define interface and behavior each relevant modules (or component).

**Description**

The interfaces between the modules are to be described to a level where an implementation is unambiguous.

The logical flow of data is to be described especially for safety relevant data.

The Software Architect must take care that the interfaces of each software module are minimal. This also corresponds to the general requirements for software architectures as described in Task 2: (e.g., maximum cohesion and a minimal coupling).

States, state transitions and activities that affect more than one module or block and their interactions are to be defined.

### Task 5: Create Safety Analysis Report

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Input | | SW Requirements Specification  SW Architecture Specification | | | | | |
| Output | | Safety Analysis Report | | | | | |
| D: |  | E: | Safety Manager | S: | SW Architect  SW Analyst | I: | SW Project Manager  SW Developer |
| D = Decision | | E = Execution | | S = Support | | I = Information | |

**Description**

A safety analysis report containing the identification or confirmation of safety-related parts of the SW is to be created which must also include the specification and verification of the efficiency of safety mechanisms.

This task can be omitted if the device under development has no ASIL classifications.

### Task 6: Analyze Dependent Failures

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Input | | SW Requirements Specification  SW Architecture Specification | | | | | |
| Output | | Dependent failures analysis report | | | | | |
| D: |  | E: | Safety Manager | S: | SW Architect  SW Analyst | I: | SW Project Manager  SW Developer |
| D = Decision | | E = Execution | | S = Support | | I = Information | |

**Description**

If the implementation of SW safety requirements relies on freedom from interference or a sufficient independence of SW components to each other an analysis of dependent failures is to be performed.

In case of a safety relevant device a control flow and a data flow analysis must be performed. A simulation of the dynamic parts must then be carried out as well and the safety manager has to ensure that such a simulation is carried out. For ASIL C and D devices a prototype generation and a formal verification are to be performed.

This task can be omitted if the device under development has no ASIL classifications.

### Task 7: Develop Detailed Design and Refine Behavior

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Input | | Software Requirements Specification  Software Architecture Specification | | | | | |
| Output | | Software Module and Unit Design | | | | | |
| D: |  | E: | SW Developer | S: | SW Architect | I: |  |
| D = Decision | | E = Execution | | S = Support | | I = Information | |

**Objective**

The goal of this task is to decompose the software architecture into a detailed design for each software module describing all software units and their interfaces and their dynamic behavior.

And evaluate the software detailed design in terms of interoperability, interaction, criticality, technical complexity, risk and testability.

**Description**

Basically, a detailed software module design consists of three parts:

* Design description of the units
* Interface description of the units
* Dynamic behavior description of the units

The design should have the following properties:

* Consistency of interfaces
* Simplicity
* Readability and comprehensibility
* Robustness
* Testability
* Suitability for software modification

If the module under development has an ASIL classification a semi-formal notation (e.g. UML/SysML) and/or a natural language shall be used for documentation.

For modules with an ASIL classifications that are configurable regard ISO-26262-6-Annex C.

### Task 8: Classification of the SW Modules and SW Units

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Input | | SW Module and Unit Design | | | | | |
| Output | | SW Module and Unit Design (refined) | | | | | |
| D: | SW Quality Planner | E: | SW Architect | S: | SW Developer  SW Analyst | I: | SW Project Manager |
| D = Decision | | E = Execution | | S = Support | | I = Information | |

**Objective**

The objective is to determine class of the software modules and software units.

**Description**

The SW Quality Planner ensures that the SW Architect classifies all software modules. The SW Developer and the SW Analyst support the SW Architect with this task.

A module classification should be performed as early as possible for time and expenditure estimation. At that stage, the module classes can, of course, be determined only roughly on the basis of the SW design. In the further course of the development process, the documentation will then be updated.

Please do not mix up the module classes as defined in this procedure with the classes defined for safety relevant functions (ASIL A-D). That is a classification of its own (see below).

The following classes exist:

* Class Low (e.g. simple module or no errors to be expected)
* Class Medium (e.g. module with average complexity)
* Class High (e.g. module with high complexity)

First, the individual units are to be rated, then the module consisting of the already rated units is classified. This order ensures that a complex module consisting of nested units can be rated ‘high’ even if the individual units have been rated ‘low’.

For the module classification, the SW Architect should take the following properties of a module into account[[2]](#footnote-2):

* The possible worst malfunction of the software module and its effect on the system
* The expected number of the lines of code (LOC), excluding comments and non-executable code (e.g. data). Preprocessor instructions and 'defines' are to be set to 'release code'
* The expected number of lines of comments in proportion to the lines of code (LOC)
* The complexity of the software module
* The expected number of changes
* The expected number of errors
* The developer’s 'qualification' (beginner, advanced, expert, ...). The assessment is the Software Project Manager’s obligation by arrangement with the team manager
* The status of the software module:
* will become standard software module
* is interrupt-dependent
* is a low-level function
* is a taken-over software module
* The expected number of the theoretical test cases (determination by means of static analysis). Note, the accuracy of estimation highly depends on the abstraction level of the detailed design (e.g., if pseudo code is used).

In the case of changes to software modules or units, the module class needs to be rated again. The module class(es) determined form the basis for the selection of necessary testing methods and depth in the subsequent testing process.

All expectations are checked during the software module test stage. If necessary the module classification is corrected.

The second classification to be performed is the ASIL classification. A module inherits the ASIL classification of the highest rated requirements it covers unless an ASIL decomposition is performed.

The third necessary classification is the origin of the module: 3rd party – off the shelf, 3rd party – contractor, inhouse, customer, generated (including by which tool). This is necessary in order to select the appropriate Q-AC subset (see MHE-PE-13).

### Task 9: Develop Verification Criteria

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Input | | SW Architecture Specification | | | | | |
| Output | | SW Architecture Specification | | | | | |
| D: |  | E: | SW Architect | S: | SW Analyst  SW Integration Tester | I: |  |
| D = Decision | | E = Execution | | S = Support | | I = Information | |

**Objective**

The objective is to define that how to validate each elements.

**Description**

A description of what must be tested to guarantee the correct fulfillment of a requirement and/or model element by the architecture must be defined and documented. A traceability to the requirement and/or model element must be given.

### Task 10: Ensure Traceability and Consistency to SW Architecture

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Input | | SW Architecture Specification  SW Module and Unit Design | | | | | |
| Output | | Traceability Record | | | | | |
| D: |  | E: | SW Developer | S: | SW Architect | I: |  |
| D = Decision | | E = Execution | | S = Support | | I = Information | |

**Objective**

The goal of this task is to ensure consistency between the software architecture and the detailed design.

**Description**

Consistency is supported by establishing and maintaining bilateral traceability between the software architecture and the detailed design.

The SW Developer must ensure traceability by using defined mechanisms.

### Task 11: Ensure Traceability and Consistency to SW Requirements

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Input | | SW Requirements Specification  SW Architecture Specification | | | | | |
| Output | | Traceability Record | | | | | |
| D: |  | E: | SW Architect | S: | SW Analyst | I: |  |
| D = Decision | | E = Execution | | S = Support | | I = Information | |

**Objective**

The goal of this task is to ensure consistency of software requirements to the software architecture.

**Description**

Consistency is supported by establishing and maintaining bilateral traceability between the software requirements and software architecture.

This step is very similar to Task 3: . It mainly serves as preparation for the verification of the software design. The Software Architect must ensure that bilateral traceability between the software architecture and the software requirements is established and the allocation is still consistent.

### Task 12: Verify SW Architecture and Module & Unit Design

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Input | | SW Architecture Specification  SW Module and Unit Design | | | | | |
| Output | | Review Checklist SW Architecture Specification  Review Checklist SW Module and Unit Design  Review Checklist Model-based Module Design  PTC-IM Review Master | | | | | |
| D: |  | E: | SW Architect  SW Developer | S: | System Architect  SW Analyst  SW Integration Tester  SW Module Tester  SW Quality Planner | I: | SW Project Manager |
| D = Decision | | E = Execution | | S = Support | | I = Information | |

**Objective**

The goal of this task is to ensure that the software architecture and the module & unit design is complete and meets all software requirements.

**Description**

This is done by a software design review, Controlling and reporting. Depending on whether the software architecture or the software module & unit design is reviewed the corresponding checklists have to be used.

It is recommended to perform a real time analysis of the software architecture in order to verify the timing behavior; data consistency and the usage of the available resources All provided measuring variables for the analysis of the qualitative and quantitative aspects of the projects and the processes are defined in the MHE-PE-05-A (Measurement and Analysis in the Development of Product-related SW).

### Task 13: Create and communicate baseline

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Input | | Software Architecture Specification  Software Module and Unit Design | | | | | |
| Output | | Software Architecture Specification (Baselined)  Software Module and Unit Design (Baselined)  Communication Record | | | | | |
| D: |  | E: | SW Architect  SW Developer | S: | Configuration Manager | I: |  |
| D = Decision | | E = Execution | | S = Support | | I = Information | |

**Objective**

The objective is to create a baseline of all documents updated or created during this process.

**Description**

The SW Architect shall initiate a baselining of all documents and ensure that this baseline and its content is known in the project team.

# Changes, References, Appendix, Terms

## References

|  |  |  |
| --- | --- | --- |
| **Category** | **Document Name** | **Document Number** |
| Process | Design of Product-related SW | AD-PE1-1-15 |

## Template

|  |  |  |
| --- | --- | --- |
| **Category** | **Document Name** | **Document Number** |
| Template | SW Architecture Specification | 8307 |
| Template | SW Module and Unit Design | 8364 |
| Template | Review Checklist SW Architecture Specification | 8348 |
| Template | Review Checklist SW Module & Unit Design | 8350 |
| Template | Review Checklist Model-based module design | 8330 |

## Abbreviation

|  |  |
| --- | --- |
| **Abbreviation** | **Description** |
| **SW PM** | Software Project Manager |
| **PM** | Project Manger |
| **SW** | Software |
| **QPS** | Software Quality Planner |
| **SYS RS** | System Requirements Specification |
| **SYS AS** | System Architecture Specification |
| **SW RS** | Software Requirements Specification |
| **SW AS** | Software Architecture Specification |

1. If reused modules are safety relevant these modules have to be qualified with respect to ISO26262. [↑](#footnote-ref-1)
2. Some of these properties are hard to estimate. During the Software Architecture Design it is, of course, not possible to elicitate these metrics automatically by use of a tool like QA-C. One possibility to improve the accuracy of estimate is to refer to similar already implemented modules (from other projects). [↑](#footnote-ref-2)