SW Detailed Design Presentation

Diplomado en Software Embebido

27 August 2021
Author.

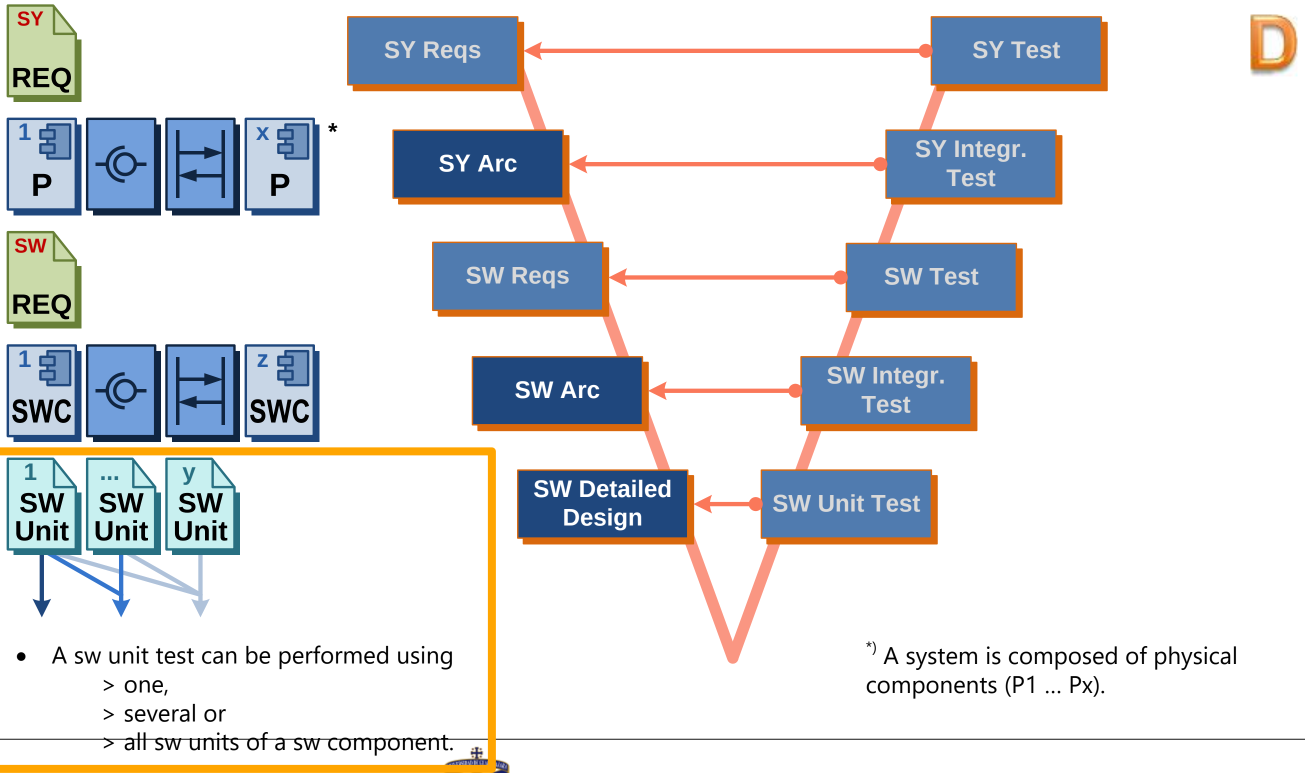
Agenda

- Introduction
  - What is SW Detailed Design?
  - Software Detailed Design on V-Cycle Process
- What are the inputs to create an SDD
- What are the parts of a SW Component?
- File Structure
- Structure for decomposition
- Traceability
- Sections on a SWDD
  - Dynamic Behavior
    - Os Task usage for this SW component
    - Interrupts
    - Error Manager
    - Power Mode management

What is SW detailed Design?

- SWDD (SW Detailed Design) is the detailed description of a SW component according to the definition on the SW architecture.
- Once a SW component boundaries, functionality, resources and requirements have been identified, the SW component is ready to be designed.
- SWDD shall describe mainly the internal behavior of a SW Component.
- A good starting point to develop a SW DD is to create an internal block diagram.
- A software component shall be designed and implemented by SW developer.

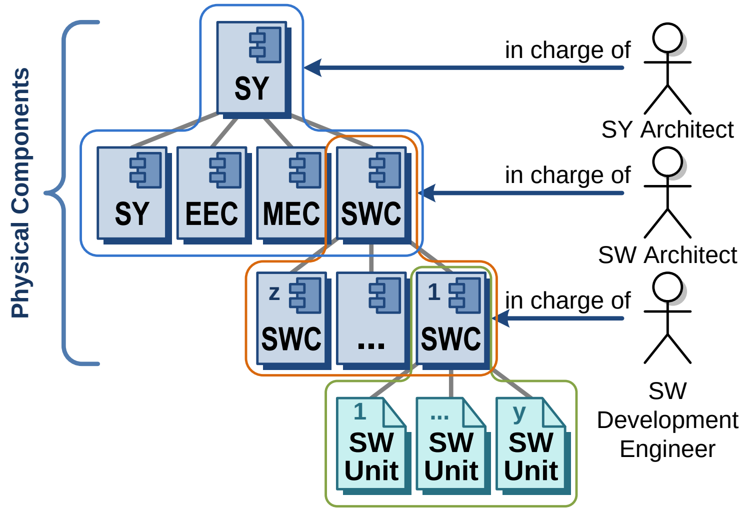
Software Detailed Design on V-Cycle Process



- A sw unit test can be performed using
  > one,
  > several or
  > all sw units of a sw component.

*) A system is composed of physical components (P1 … Px).

Architectural Decomposition view



SY Architecture
The sy architecture recursively decomposes a system into smaller physical components.

SW Architecture
The sw architecture recursively decomposes a sw component into smaller sw components.

SW Detailed Design
The sw detailed design decomposes a sw component into sw units.
SW Unit = SW Modul = ~.c File

Architectural Decomposition

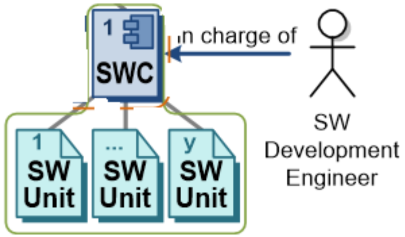
- An architecture consists of physical components that can be further decomposed into more fine-grained physical components across appropriate hierarchical levels.
- The software components are the lowest-level physical components of the software architecture for which finally the detailed design is defined.
- A software component consists of one or more software units.

What are the inputs to create an SDD

- Before to start working on SW detailed design those workproduct shall have sufficient level of maturity.
  - SW requirements
  - SW Architecture.
- If the workproducts listed above has not mature enough or not reviewed
- Based on SW Requiremens Allocation from SW Architecture. SW developer can start working on SDD using this information.

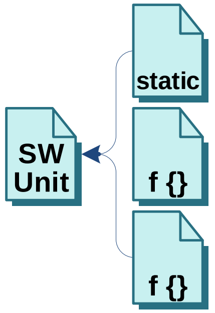
What are the parts of a SW component?

- As it is described on the Decomposition view SW C are decomposed into sw units.
- As a practical way let's consider a *.c file as a SW unit.



SW Detailed Design
The sw detailed design decomposes a sw component into sw units.
SW Unit = SW Modul = ~.c File

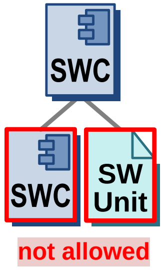
What are the parts of a SW component?



- A sw unit is the smallest possible compilable sw building block.
  > in C++ it is a class
  > in C it is a ~.c file with corresponding ~.h file (optional)
- It contains operations (C-functions) and data to realize an as far as possible self-contained job.
- There can also be other criteria for creating sw units, e.g. organisational constraints.
- The cohesion within a sw unit should be high.

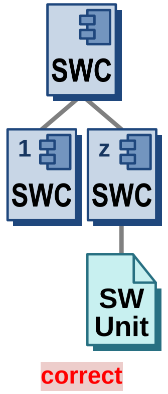
SW Unit = SW Modul = ~.c File

What are the parts of a SW component?



Rule:

- A sw component has to be broken down into either sw components or sw units.



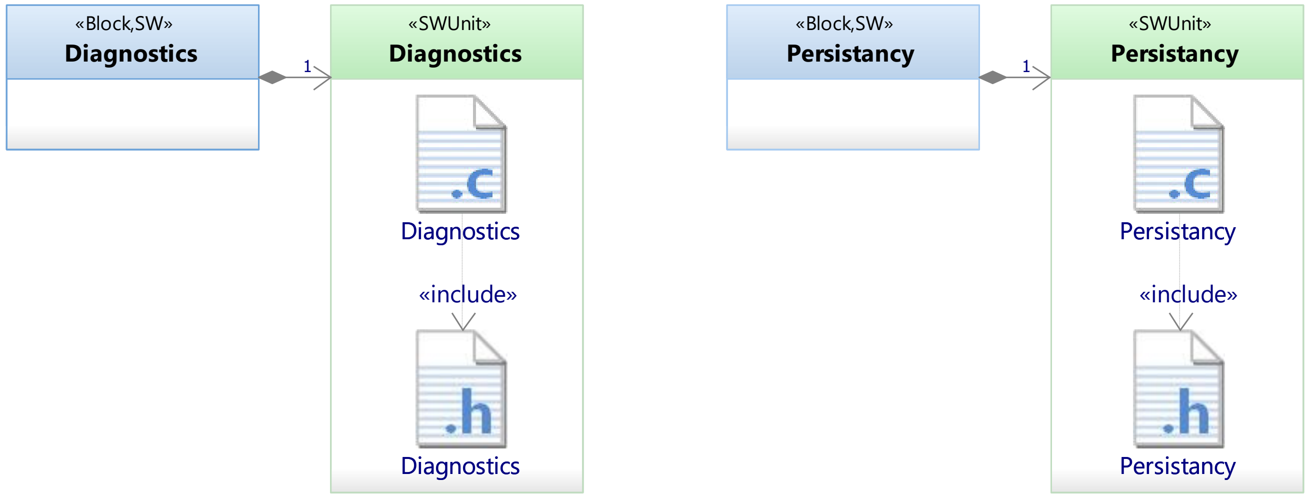
File Structure

- In order to get a consistent structure for all sw detailed designs the sw architecture has to define the overall file structure for a sw detailed design.

- This structure shall consider how for instance the following content is handled within the SWDD:
  - header files which provide the public types and interface(s).
  - header files which provide private types and interface(s) and shall never be included by other sw components as this internal interfaces may change anytime without notice.

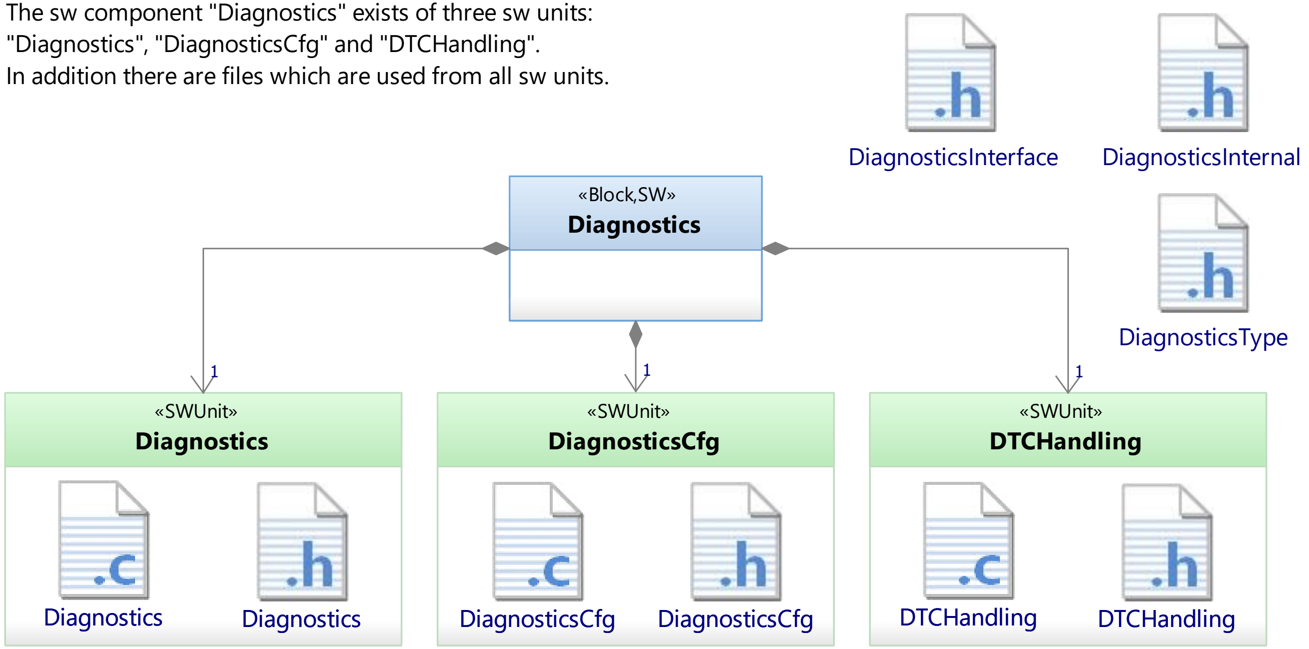
SWUnit • Naming Rules • Example

Each sw component is represented by a sw unit with the same name. As a sw unit is a ~.c file with corresponding ~.h file these files have the name of that sw component / sw unit.



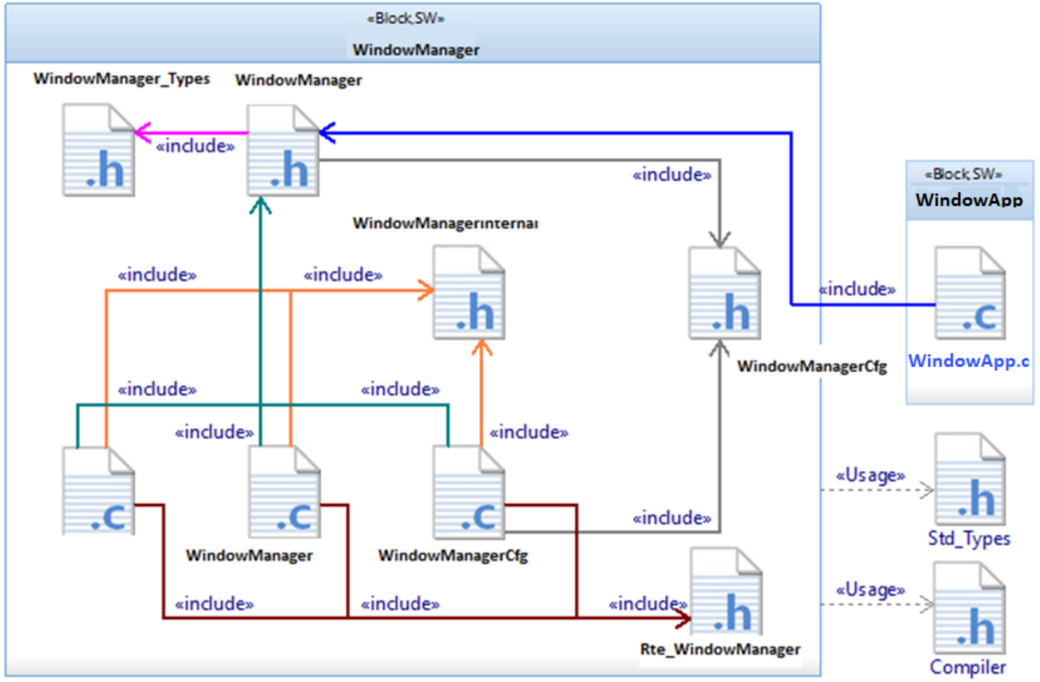
SWDD • Decomposition • Example

The sw component "Diagnostics" exists of three sw units:
"Diagnostics", "DiagnosticsCfg" and "DTCHandling".
In addition there are files which are used from all sw units.



CUCEI

SW Units example for Window Manager



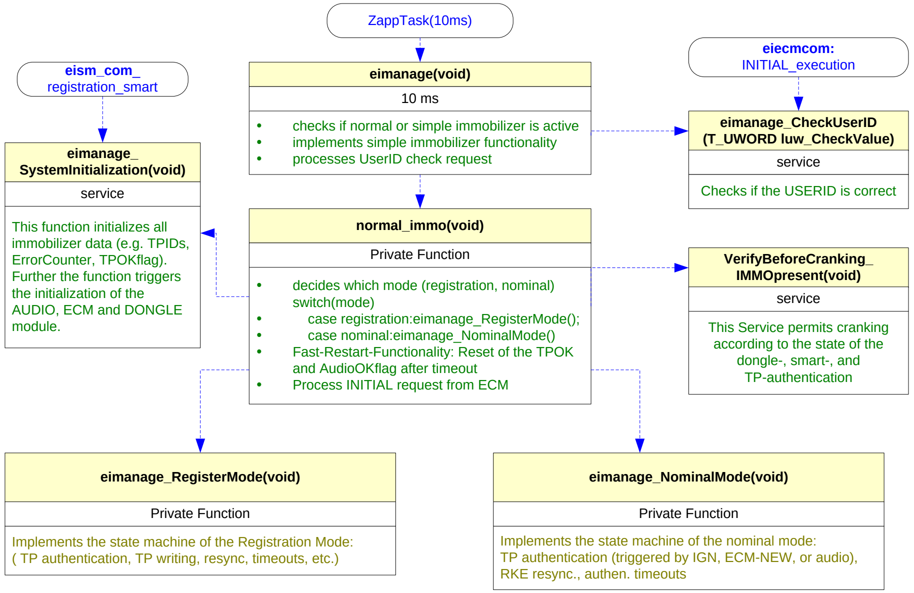
CUCEI

Sections on SDD

- Those are the basic sections included on a SDD document:
- Functional Decomposition
- Requirements allocation
    - Unit Level
    - Function Level
- Operations
    - Initialization / Shutdown functions
    - Os Task Functions
    - Isr Functions
    - Interfaces
    - Private Operations
- Global variables
- Macros (Reelevant Ones)
- Algorithms
- Shared Resources Analysis
- Resource Consumption

Sections on SDD - Functional Decomposition

- In a similar approach that SW architecture it is important to identify the interaction among different internal parts of SWC.
  - Variables
  - Operations
- As it was mentioned before the easiest way to represent this is using an internal block diagram. Also, an activity diagram could work to reflect this information as second option.



ZappTask(10ms)
eism_com_ registration_smart
eiecmcom: INITIAL_execution
eimanage(void)
10 ms
checks if normal or simple immobilizer is active
implements simple immobilizer functionality
processes UserID check request
eimanage_ SystemInitialization(void)
service
This function initializes all immobilizer data (e.g. TPIDs, ErrorCounter, TPOKflag). Further the function triggers the initialization of the AUDIO, ECM and DONGLE module.
eimanage_CheckUserID (T_UWORD luw_CheckValue)
service
Checks if the USERID is correct
normal_immo(void)
Private Function
decides which mode (registration, nominal) switch(mode)
case registration:eimanage_RegisterMode();
case nominal:eimanage_NominalMode()
Fast-Restart-Functionality: Reset of the TPOK and AudioOKflag after timeout
Process INITIAL request from ECM
VerifyBeforeCranking_ IMMOpresent(void)
service
This Service permits cranking according to the state of the dongle-, smart-, and TP-authentication
eimanage_RegisterMode(void)
Private Function
Implements the state machine of the Registration Mode: (TP authentication, TP writing, resync, timeouts, etc.)
eimanage_NominalMode(void)
Private Function
Implements the state machine of the nominal mode: TP authentication (triggered by IGN, ECM-NEW, or audio), RKE resync., authen. timeouts

Sections on SDD - Requirements Allocation

- As it was mentioned before SWA will indicate requirements allocation at component level. It means the requirements that the SWC needs to satisfy.
- On a recursive way a new decomposition will occur at SW Component level (as it was mentioned before). A SWC can be decompose in:
  - Other SWC (Optional, size of SWC need to be analyzed)
  - Units (At least 1)
- Recursively Units will decompose into:
  - C Functions

Sections on SDD - Requirements Allocation

- In order to have a proper traceability requirements allocation will be performed at
  - C File
  - Function Level
- In this way it will be easier to identify in which function a requirement has been implemented.
- This will help to identify initial points for debugging when an issue is found.

Sections on SDD - Operations

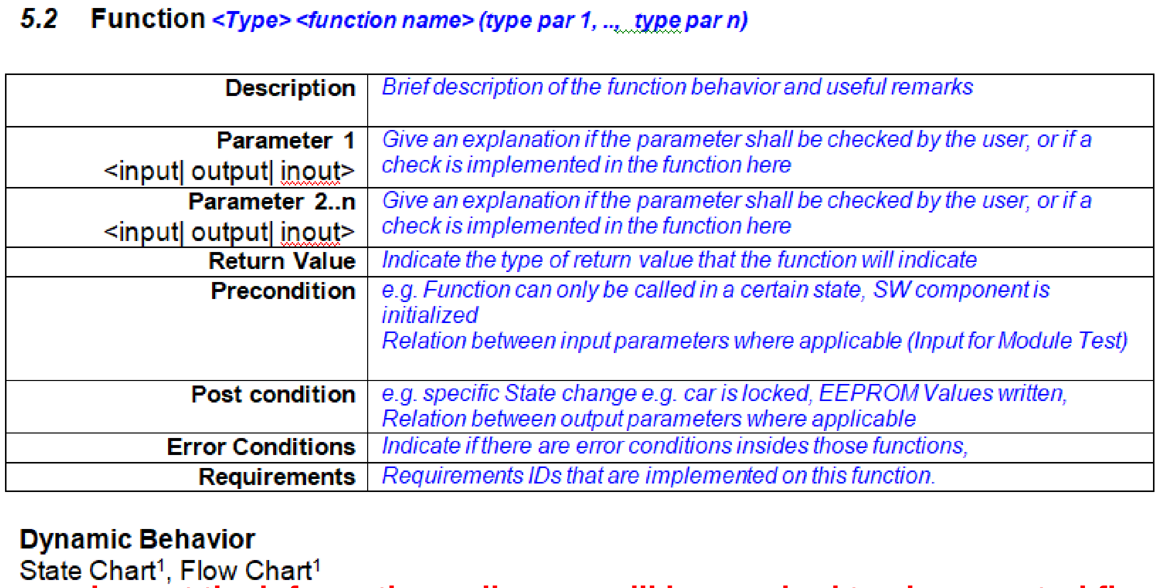
› Operations are the essence of the source code. a.k.a. C functions they will indicate the logical behavior that the SW Component will implement.

› There are different types of operations. Relevant ones are:

Function Type	Description
Initialization function	Used to indicate configurations when the SWC initialize. Typically variables take initial value or No Init Variables are handle.
Shutdown functions	Used to indicate configurations when the SWC goes to shutdown mode (sleep mode) are used to actiave process that will be executed on shutdown but also to stop other process that are no longer required to execute during shutdown
Os Task Functions	Used to indicate small Portions of the SWC functionalities that are called by the OS. Main types of those at application level are: Synchronous: Executed with a fixed period. Asynchronous: Executed under task activation request.
Isr Functions	Functions executed during a Interrupt execution. Recommendation is that those functions fast as possible.
Interfaces	Functions used to exchange information with other SWC. Main types are •Data Input: To request a Data value from other SWC. •Data Output: To provide a Data value to other SWC. •Event Input: Notification that a event has been occurred. (Callback) •Event Output: Notification to other SWC that an event has been occurred.
Private Operations	Operations implemented internally on the SWC that are no relevant to other SWC but there are part of the implementation. a.k.a as static operations.

Sections on SDD - Operations

- For every operation it is required to describe:



5.2 Function *<Type> <function name> (type par 1, ..., type par n)*

Description	*Brief description of the function behavior and useful remarks*
Parameter 1 <input| output| inout>	*Give an explanation if the parameter shall be checked by the user, or if a check is implemented in the function here*
Parameter 2..n <input| output| inout>	*Give an explanation if the parameter shall be checked by the user, or if a check is implemented in the function here*
Return Value	*Indicate the type of return value that the function will indicate*
Precondition	*e.g. Function can only be called in a certain state, SW component is initialized* *Relation between input parameters where applicable (Input for Module Test)*
Post condition	*e.g. specific State change e.g. car is locked, EEPROM Values written,* *Relation between output parameters where applicable*
Error Conditions	*Indicate if there are error conditions insides those functions,*
Requirements	*Requirements IDs that are implemented on this function.*

Dynamic Behavior
State Chart[1], Flow Chart[1]

- **In order to complement the information a diagram will be required to show control flow and data flow. The most used for this are Activity Diagrams or State Machines.**

Sections on SDD – Global Variables

- Identify and Describe what will be the global variables that will be defined inside this SWC.
- Identify and describe the Data Types that will be defined by this SWC.
- A good description of them will help the developers to simplify the implementation.

Sections on SDD – Macros

- Describe the macros that will be used in the code will relevant information.
- Examples of relevant macros:
  - SW Configuration Macros (Threshold values, Loop operation control values, Buffer size definition, Pointer aritmethic value macros)
  - Precompile Switches configuration
  - Functions Like Macros (Those are not recommended by MISRA Rules)

Sections on SDD - Algorithms

- Most of the time, Application SWC implements specific functionalities that requires a specific algorithm.
- Those algorithms needs to be described on SDD in order to indicate a reference for the developer.
  - Examples:
    - Algorithm to detect a anti pinch event sensorless.
    - Algorithm to find a key outside a vehicle.
    - Algorithm to brake with an ABS module.

Sections on SDD - Shared Resources Analysis

- In order to indicate that there won’t be conflict for shared resources. An analysis needs to be performed at SWC level. For this analysis the Global Variables documetned on this SDD needs to be considered if:
  - They are used on more that one Os Task.
  - They are shared and modifiable via interface by other SWC.
  - They are modified inside ISR.
- If this is the case an analiysis for Shared resources needs to be performed and mechanisms for sharing data needs to be implemented.
  - Reentrant code
  - Mutex
  - Semaphores
  - Atomic code

Sections on SDD - Resource Consumption

- SDD need to indicate on a more accurate way the resource consumption estimation that the one performed at SWA level.
- This is because SDD will contain more details about the implementation for the SWC.
- This estimation will reflect:
  - RAM memory consumption
  - ROM memory consumption
  - CPU Load
  - Microcontroller resources

Exercise

- Read Example and create a design proposal for SWC for Window Control Driver. This Driver is responsible to control the operations of the window. Mainly it will operate the H Bridge and report Window Position.
  - Operations for this SWC:
    - Determine window Position
      - Provide Switch Position
    - Provide services to operate window
      - Open window
      - Close window
      - Cancel Window Operation
      - Provide Window Position
    - Execute HW diagnostics
      - Diagnose Errors
      - Report Errors to application
    - Configure HW diagnostics

- Interfaces
- Window Position
- WINDOW_POSITION WindowDrvr_Get_WindowPosition();

- Window Control
- WINDOW_ERR WindowDrvr_CloseWindow()
- WINDOW_ERR WindowDrvr_OpenWindow()
- WINDOW_ERR WindowDrvr_WindowUp()
- WINDOW_ERR WindowDrvr_WindowDown()
- WINDOW_ERR WindowDrvr_WindowCancel()

- DTC Callback
- WindowDrvr_WindowDiagPASS_Cbk()
- WidnowDrvr_WindowDiagFAIL_Cbk()
-
-
- DID$0195
- WindowDrvr_WindowSetDiagCfg(uint16 OpenWindowDiagMinThr,
- Uint16 OpenWindowDiagMaxThr,
- Uint16 CloseWindowDiagMinThr,
- Uint16 CloseWindowDiagMaxThr,
- Unit16 IdleDiagMinThr,
- Uint16 IdleDiagMaxThr)
- WindowDrvr_WindowGetDiagCfg(uint16* OpenWindowDiagMinThr,
- Uint16* OpenWindowDiagMaxThr,
- Uint16* CloseWindowDiagMinThr,
- Uint16* CloseWindowDiagMaxThr,
- Unit16* IdleDiagMinThr,
- Uint16* IdleDiagMaxThr)

- H BRIDGE INTERFACE
- Dio_WriteDirect(TRANSISTOR_DC_1,1)

- Dio_WriteDirect(TRANSISTOR_DC_2,0)

- Dio_WriteDirect(TRANSISTOR_DC_3,0)

- Dio_WriteDirect(TRANSISTOR_DC_4,1)

- Adc_Read(Chanel)

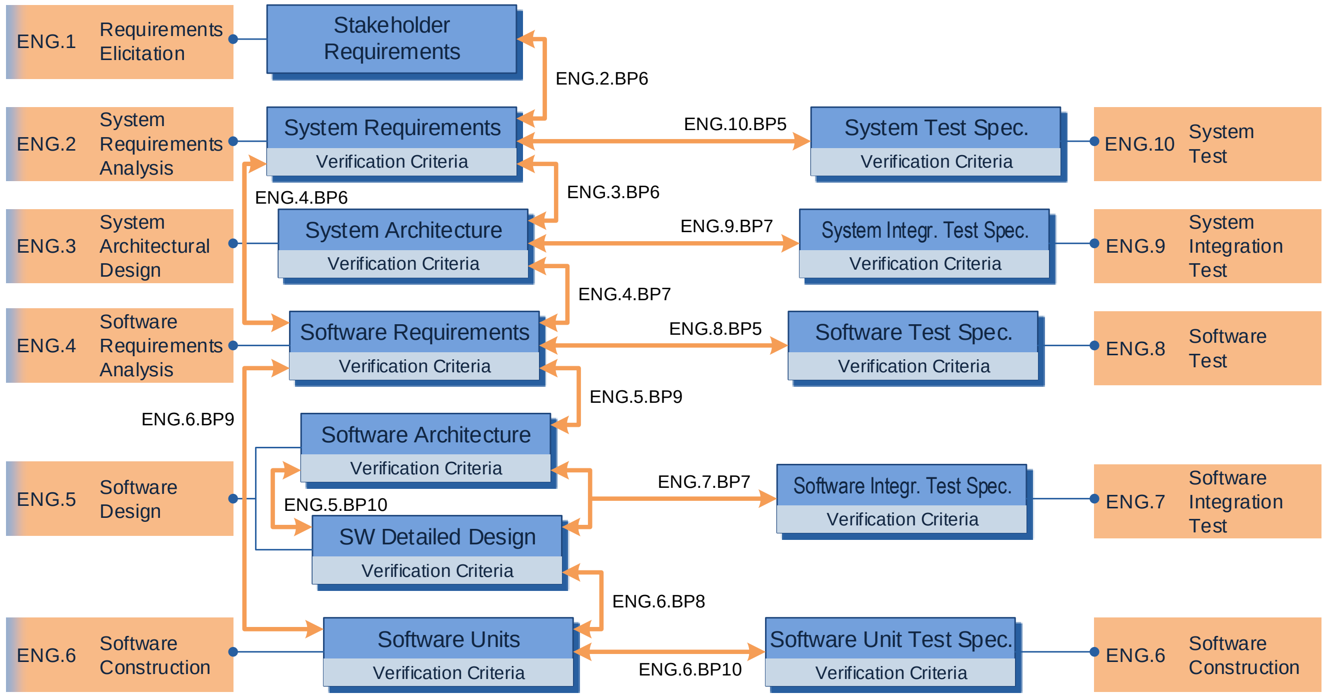
- 3 C files
- WindowDrvr_Control.c/.h
- WindowDrvr_HwDiag.c/.h
- WindowDrvr_WindowPosition.c/.h
- WindowDrvr_HwDiagCbk.h

CUCEI

Exercise

- How Many Units are required to have a proper implementation?
- How many OS tasks will be required to perform the implementation?
  - How many will be periodic?
  - How many will be per request?
- Do you identify usage of ISR?
- Do you identify usage of other microcontroller resources?

Traceability on V Cycle Process

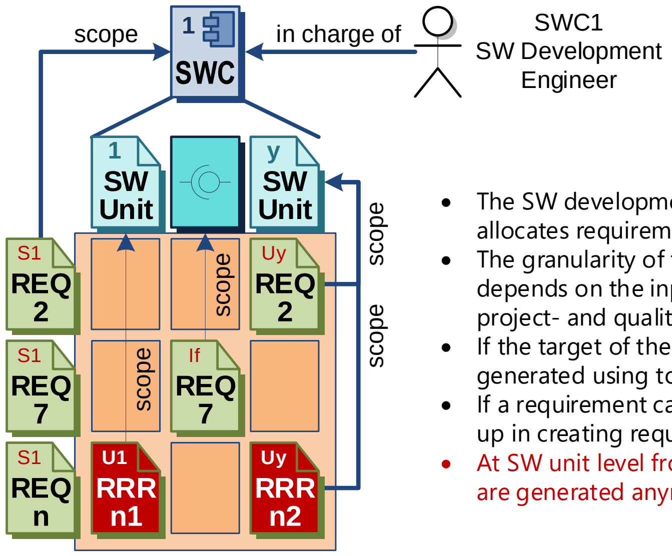


Overview of required cross-references according Automotive SPICE, Process Assessment Model, Version 2.5 (PAM 2.5)

Traceability

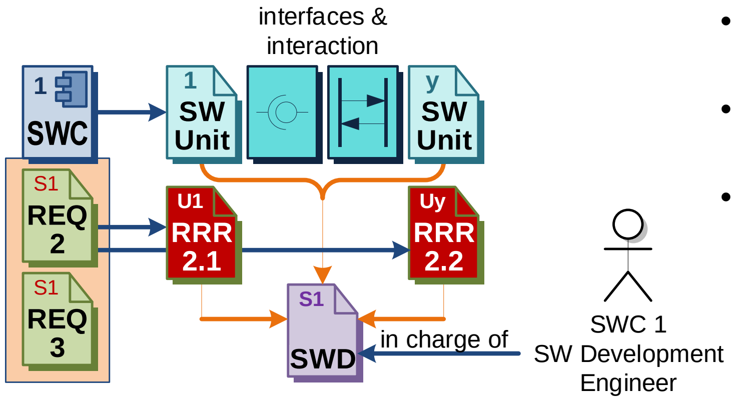
- Traceability does not necessarily mean to implement explicit links (e.g., <<satisfy>>, <<trace>>, etc.).
- It is also allowed to use other mechanisms to get the trace path, for example if elements are connected via name.
- If source code generation is not used, traceability via name is the preferred way to trace from sw detailed design to source code.
- The granularity of traceability - whether to allocate to sw units, (sw component) interfaces or operations, depends on the input of the customer and has to be discussed with the project- and quality-management.

Requirement Allocation & Refinement • SW Unit Level



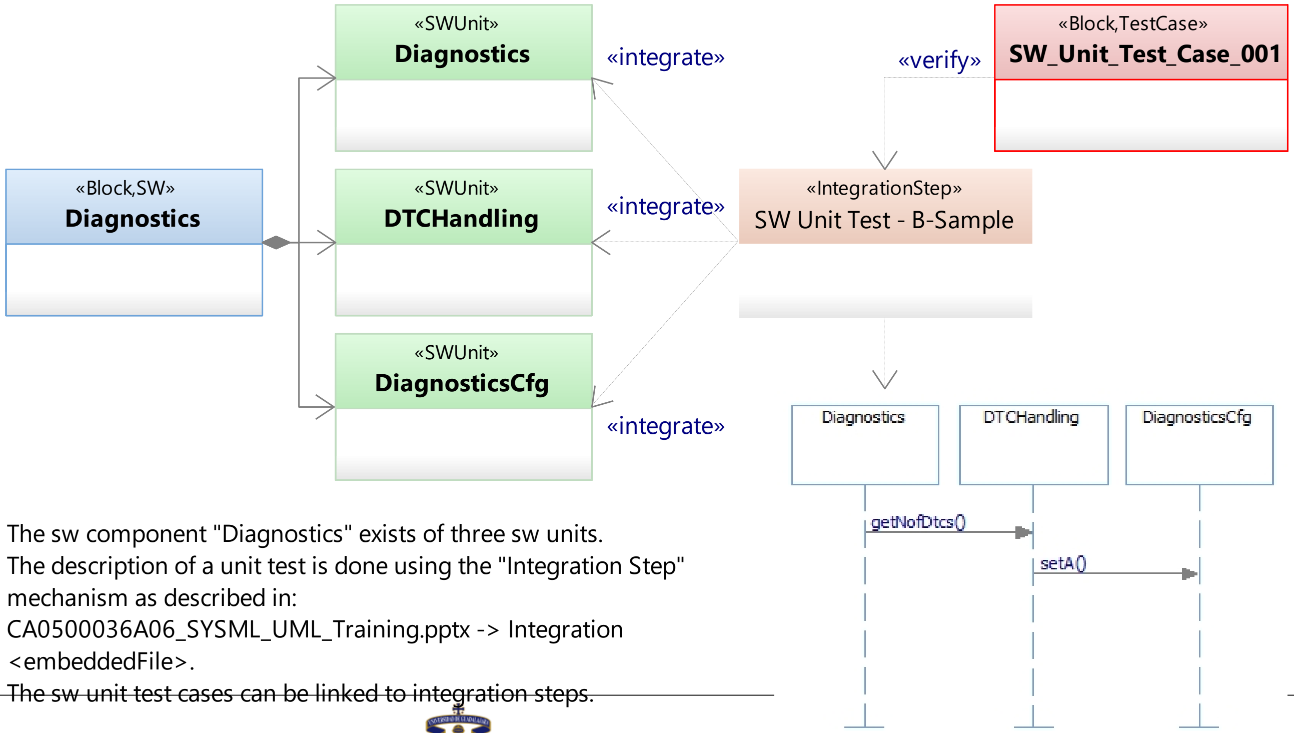
- The SW development engineer, who is in charge of the SW component SWC1, allocates requirement REQ2 to SW unit y and REQ7 to an interface.
- The granularity of traceability – whether to allocate to SW units or interfaces – depends on the input of the customer and has to be discussed with the project- and quality-management.
- If the target of the allocation are interfaces, the subsequent traceability can be generated using tools, which are able to generate caller graphs.
- If a requirement cannot be assigned exclusively to a sw unit, it has to be split up in creating requirement refinement rationales (RRR).
- At SW unit level from requirement refinement rationales no SW requirements are generated anymore!

SW Detailed Design



- SW component 1 is not decomposed into smaller sw components, but decomposed into SW units.
- From SW requirement refinement rationales, which are assigned to SW units, no requirements are generated.
- The assigned requirement refinement rationales and the SW unit decomposition – inclusive interfaces and interactions, are part of the SW detailed design for that SW component.

Traceability • SW Detailed Design - SW Unit Test



The sw component "Diagnostics" exists of three sw units.
The description of a unit test is done using the "Integration Step" mechanism as described in:
CA0500036A06_SYSML_UML_Training.pptx -> Integration <embeddedFile>.
The sw unit test cases can be linked to integration steps.

End of session