MCAL Development Overview

Tu Pham

## Content

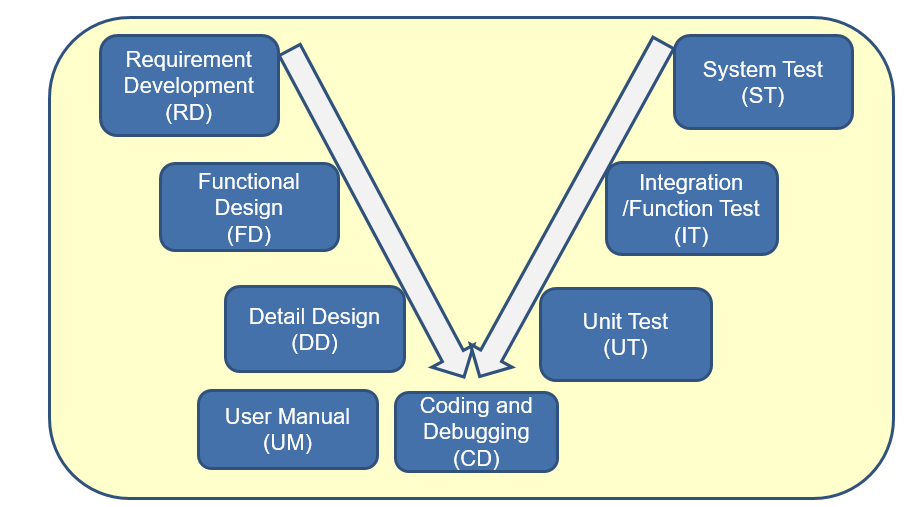
* Requirement development
  + Hardware user's manual/Operation pre-cautions/Errata
  + JIRA/Customer requirement
  + SWS/IA/MRS
* Functional design
  + FD/DD template/guideline
* Detailed design
  + DD template/guideline
* Coding and debugging
  + ECU configuration (PDF, BSWMDT, CDF)
  + C/C#/ARXML language, IDEs (Notepad++, MULTI, VS Pro), and coding rule
  + Static analysis (Metrics, Polyspace Bug Finder, QAC, AMDC, Coverity)
* Writing MCAL User Manuals
  + UM template/guideline (Common, Generation Tool, Driver)

## Objective

Trainees should be able to introduce concepts:

* V - model
* RVC Work Products corresponding to V-model
* FD: Configuration, Error List
* DD: Document Detail
* TCODE
* Architecture TCODE, ECODE
* Static Test
* IT, UT
* Why do we need to review activities?
* What is peer review?

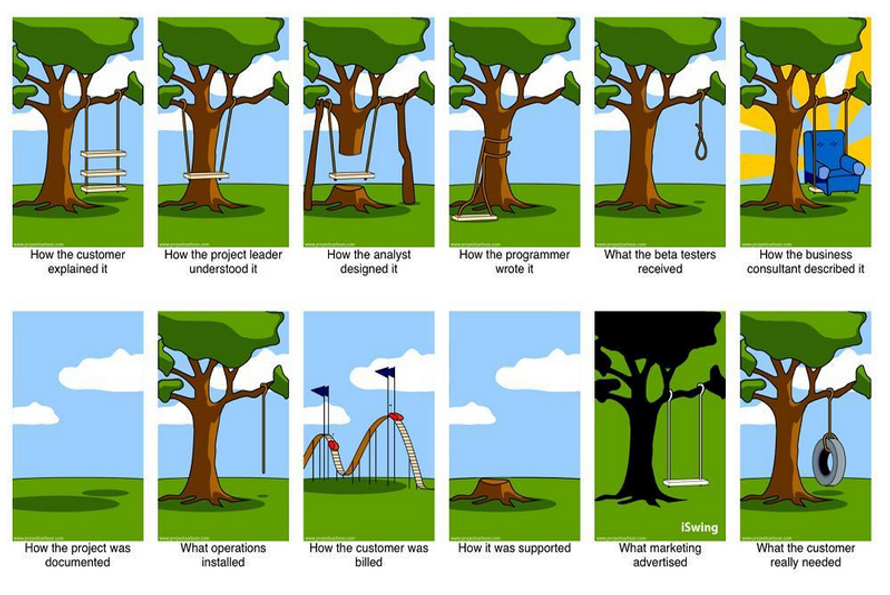
### The RVC MCAL Software development in V-Model process that is described as below:



### Requirement Development (RD)

Why is RD necessary?

* Implicit assumption for requirement is one of the key reasons to project and software failures.
* Developers doesn’t know what is considered “Complete”.
* Testers must make assumptions and spend time defining or looking for hidden requirements themselves.
* It’s not uncommon for bugs to be introduced due to unclear requirements or a misunderstanding of them.



What is RD?

* Requirement development is to convert customer's requests, market demands, and internal requests into the functions that are incorporated in the software product and obtain agreement from the sources of the requests about the requirements.
* In the Requirement Development Process, requirement development is implemented, and the Requirement Specification is created in accordance with the Project Plan (Basic Information, Management Information, Project goals/policies).

How to do RD?

* Elicit requirements: Determine whether the stated requirements are unclear, incomplete, ambiguous, or contradictory with, and then resolve these issues by being curious as such kind of questions.
  + Is any other existing requirement related to the issue or a new requirement?
  + What is wrong/expectation behavior? Is any violation with AUTOSAR spec and HWM?
  + What is problem occurrence condition?
  + What is root cause?
* Check Impact work products
  + Analyze what work products (MRS, FD, DD, CD) will be impacted.
  + Analyze how issue impacts to the work products. E.g. New API will be added.
* RD Review
  + Team members reflect on what happened in the iteration and identifies actions for improvement going forward.

There are work products in RD as following:

* Impact analysis template and example
* Ver4.xx.xx\_**MSN**\_Impact\_Analysis\_Template.xlsm
* MRS
* Generator Requirement
* General Requirement
* Module Requirement
* Coding Guideline

**MSN** are 14 modules MCAL in RVC developments such as: ADC, CAN, DIO, ETH, FR, FLS, GPT, LIN, MCU, PORT, PWM, ICU, SPI, WDG.

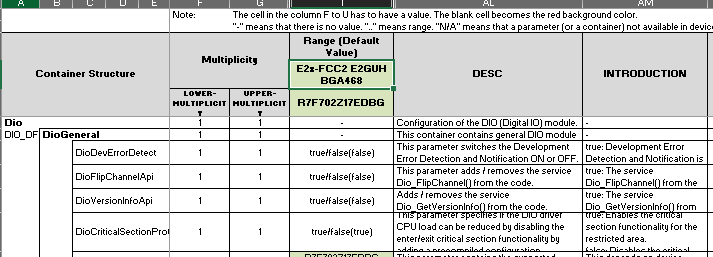
1. Functional design (FD)

Functional designing is to define the composition of the software component, to assign the requirements defined in the Requirement Specification to software components and define the interfaces between the software components.

In the Function Design Process, functional designing is implemented, and the Functional Design Document is created in accordance with the Requirement Specification formulated in the Requirement Development Process.

In RVC, FD include work products such as:

* 1. Designing for PDF file, named file as RH850\_< Platform Name>\_<MSN>\_ParameterDefinition.xlsx



* 1. Designing for generated source code by Generation Tool, named file as RH850\_<Platform Name>\_<MSN>\_ Configuration.xlsx
  2. Designing for module specific error/warning/info is used by Generation Tool, named file as RH850\_<Platform Name>\_<MSN>\_ GenTool\_ErrorList.xlsx
  3. Designing functions for Driver code, named file as RH850\_<Platform Name>\_<MSN>\_ Driver\_FD.docx
  4. Sequence diagram for Module APIs or data flow for specific module feature, named file as RH850\_X2x\_<MSN>\_SequenceDiagram.xlsx

1. Detail design (DD)

What is DD?

Detail designing is to define the internal behavior and to assign the design contents of software component defined in the Functional Design Document to software units and to define the interfaces between the software units and the processing logics of the software units at levels that enables coding and the design of the Unit Test Specification.

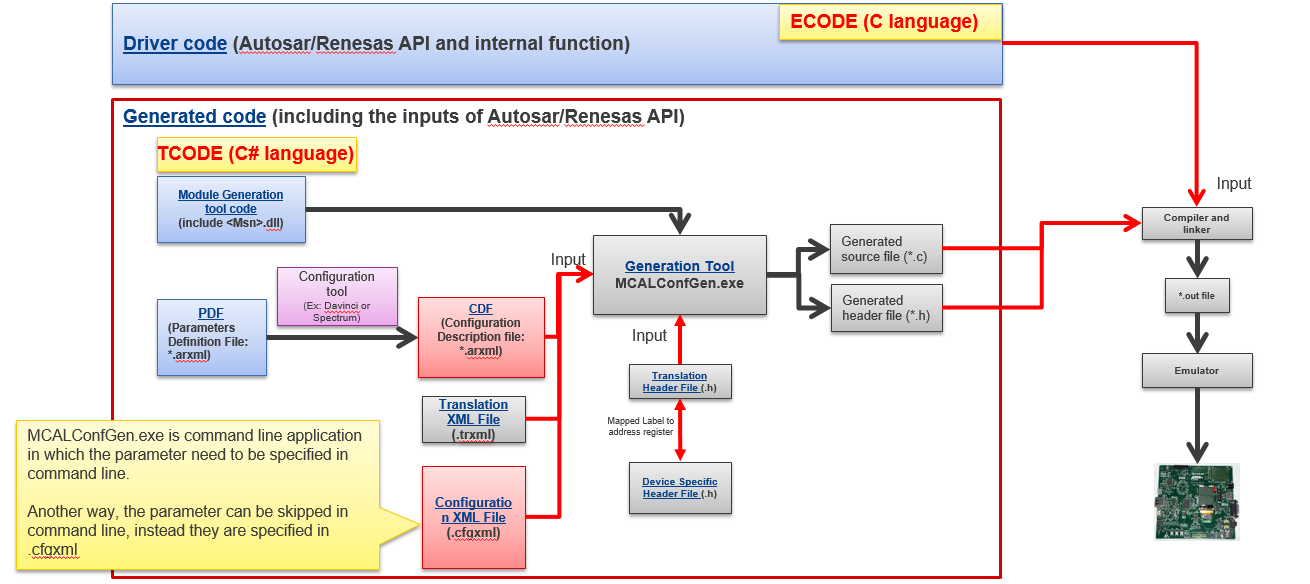
In the Detail Design Process, detail designing is implemented, and the Detail Design Document is created in accordance with the Functional Design Document formulated in the Functional Design Process

**In RVC, DD work products include General DD, Module DD. General DD is detail design for common source in generation tool, driver** while **module DD is detail design for module specific source in generation tool and detail design for module driver source like activity diagram**

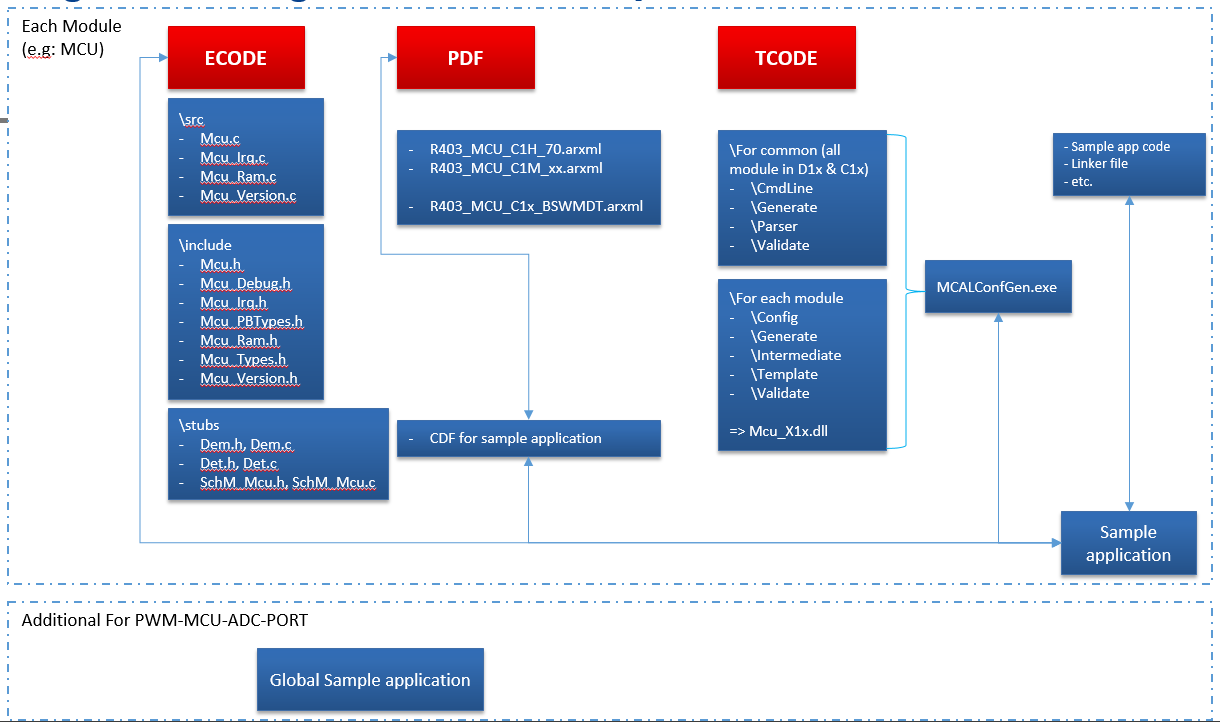
Named file of general DD as **RH850\_<Platform>\_Generic\_Gentool\_DD.docx, module DD as RH850\_<Platform>\_<MSN>\_GenTool\_DD.docx and RH850 \_<Platform>\_<MSN>\_Driver\_DD.xlsx**

1. Coding and Debug Process

In RVC, coding and debug process on two main products such as source code for Generation Tool (TCODE) and source code for Driver (ECODE).

TCODE is developed by C# language and C language is used in ECODE development. The Generation Tool is command line application in which the parameter need to be specified in command line. It is used to generate configuration by user and combine with Driver source code to compile to BOARD target. To be more detail as below:

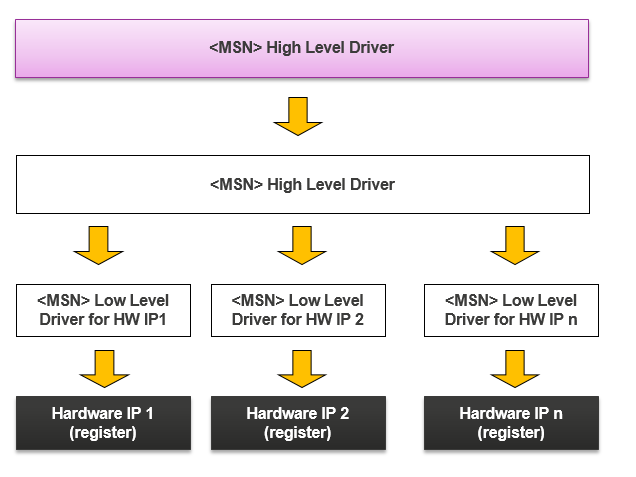
An example for MCU work product modules:



* Some work products in CD phase:
  + Driver code (C language): functions/APIs to implement the MCAL modules
  + Tool code (C# language): this code will generate source files and header files (C language) from the input files (CDF, translation file…), those generated files will store data, variables or macros which configured in CDF, and those generated files will be compiled with Driver code to implement the MCAL modules
  + PDF (arxml type): consists of all parameters of a module, main parts in the PDF is container (store many parameters), parameters, values of parameter (a parameter will have many values). CDF (arxml type) is file to store parameters and their selected value in PDF by use generation tool (Davinci, Spectrum…).
  + Sample application: consist of CDF, linker files, sample test program (\*.c, \*.h) for user, they are used to compile with work products of CD to confirm the implementation of driver
  + Stub files: include source files, header files or arxml files of other modules (not MCAL modules) but they are necessary for MCAL driver modules
  + BSWMDT (arxml type): An XML format file which contains overview information about MCAL modules such as list of API and its properties (Name, ID), memory map, … in a standard format defined by AUTOSAR (This file was used for upper layer to integrate MCAL module)
* Consist of many work products to implement a driver:

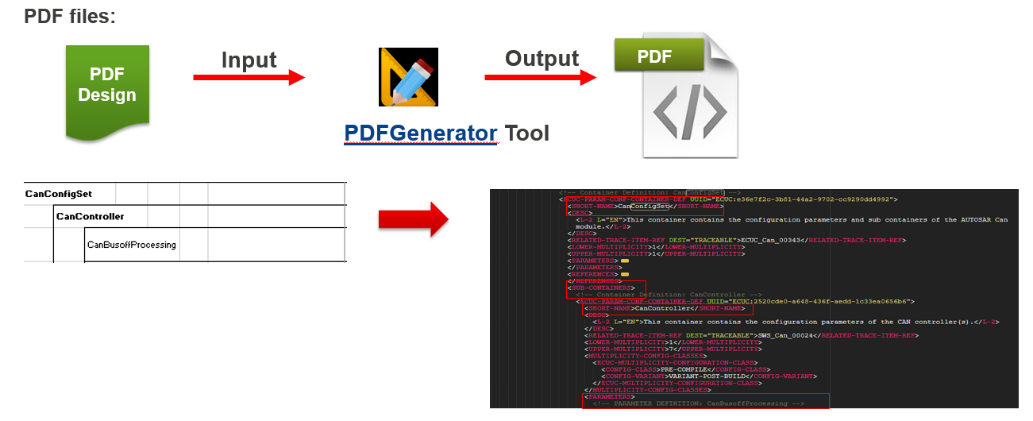
|  |  |  |
| --- | --- | --- |
| **Main Item** | **Location** | **Remark** |
| ECODE | \external\X1X\D1x\modules\<msn>\include  \external\X1X\D1x\modules\<msn>\src | MSN Driver code |
| \external\X1X\common\_platform\modules\<msn>\include  \external\X1X\common\_platform\modules\<msn>\src |
| TCODE | \internal\X1X\common\_platform\generic\generator\_cs | MSN Generation tool code |
| \internal\X1X\common\_platform\modules\<msn>\generator\_cs  Or \internal\X1X\D1x\modules\<msn>\generator\_cs |
| PDF | \external\X1X\D1x\modules\<msn>\definition | Parameter Definition File |
| BSWMDT | \external\X1X\D1x\modules\<msn>\generator | \*.arxml files |
| Stubs | \external\X1x\common\generic\stubs | - |

* Driver Architecture

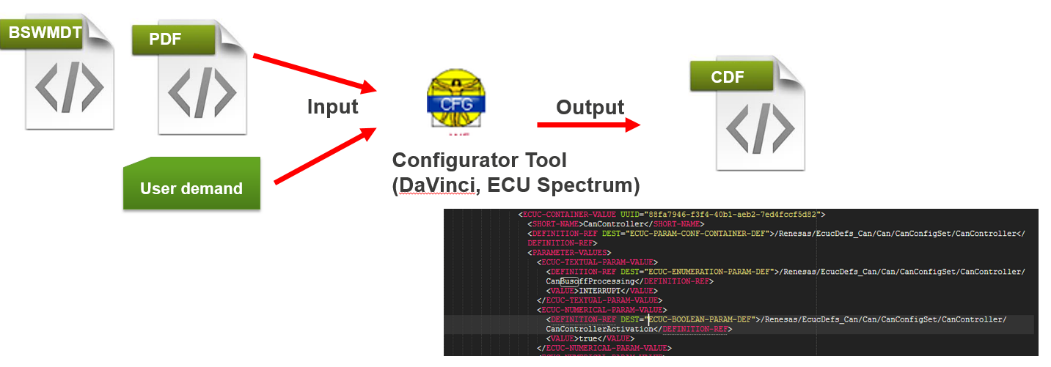


* Explanation ARXML files (PDF/CDF/BSWMDT)

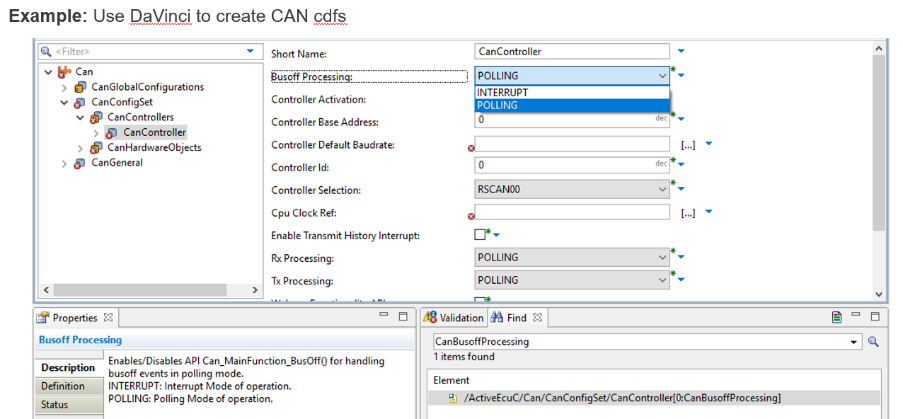
**PDF** file is described as above part that is parsed to file .arxml by tools. File pdf .arxml is XML format in Autosar standard. It was generated from PDF design. AUTOSAR standard was defined in “AUTOSAR\_TPS\_ECUConfiguration.pdf”



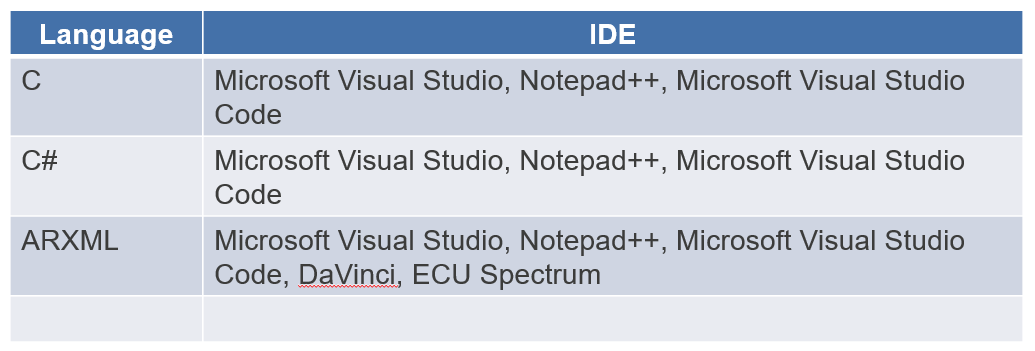
**CDF** is file used to store some parameters that is configured by User in PDF files. So, CDF is generated by PDF via tools.



Below is an example for creating CDF using tool DaVinci



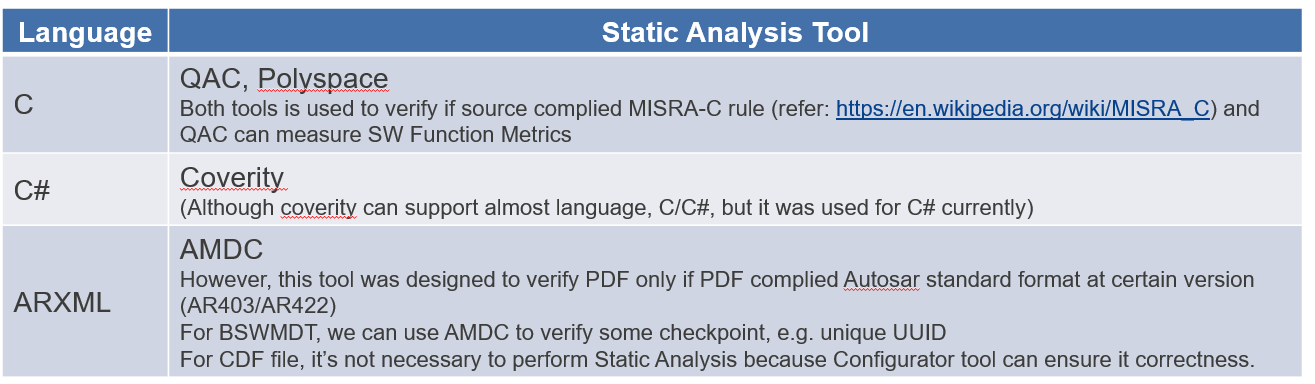
**Tools and IDE** is used to develop word product in coding phase as following:



**Static Analysis**

Static program analysis is the analysis of computer software that is performed without actually executing programs, in contrast with dynamic analysis, which is analysis performed on programs while they are executing.

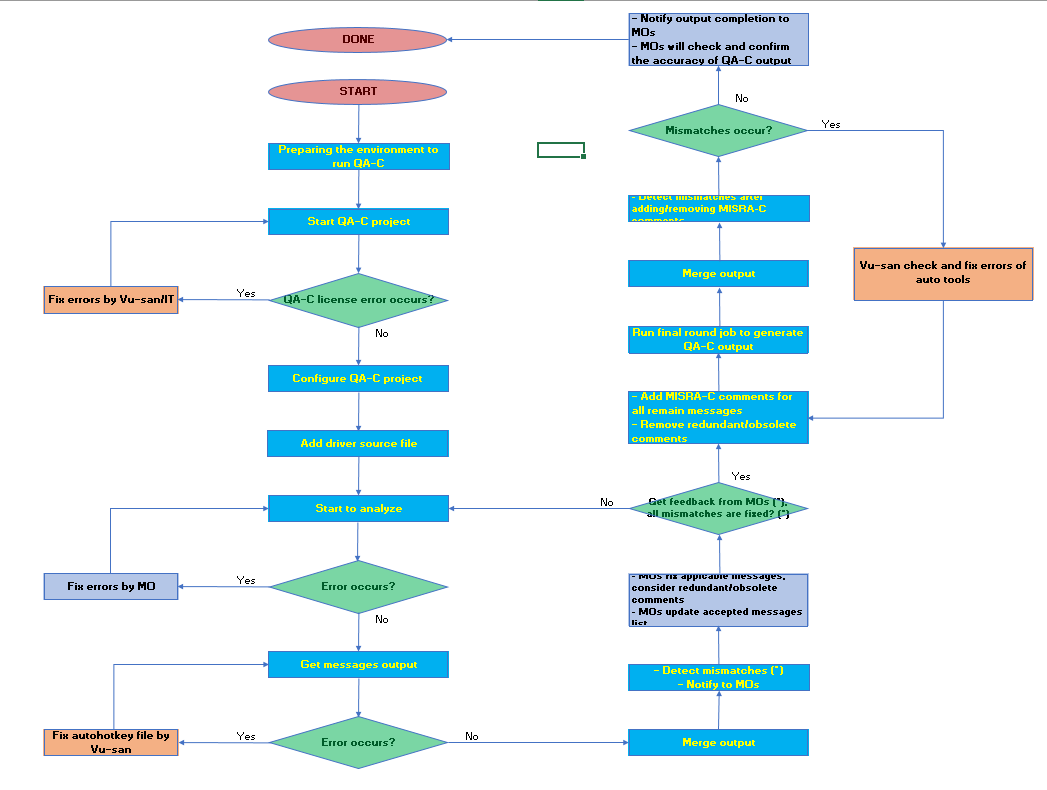
Static Analysis Tool: <https://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis>



At RVC, Static Analysis will be done by Jenkins, we can trigger corresponding jobs at http://172.29.145.61:8080/

After Jenkins finish analysis, a notification with attached report will be sent via email.

The workflow for QAC jobs was attached as below:



User Manual

UM = user manual. The UM will guide user basic knowledge to use a driver of module such as API information, feature/ functionality of module, error or warning message. There are 2 UM documents: one for driver information (ECODE), one for generation tool information (TCODE).

Example:

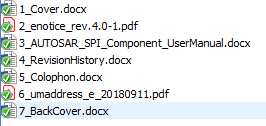
**Location:**

**Develop version:** \internal\X1X\D1x\modules\<msn>\docs\user\_manual\

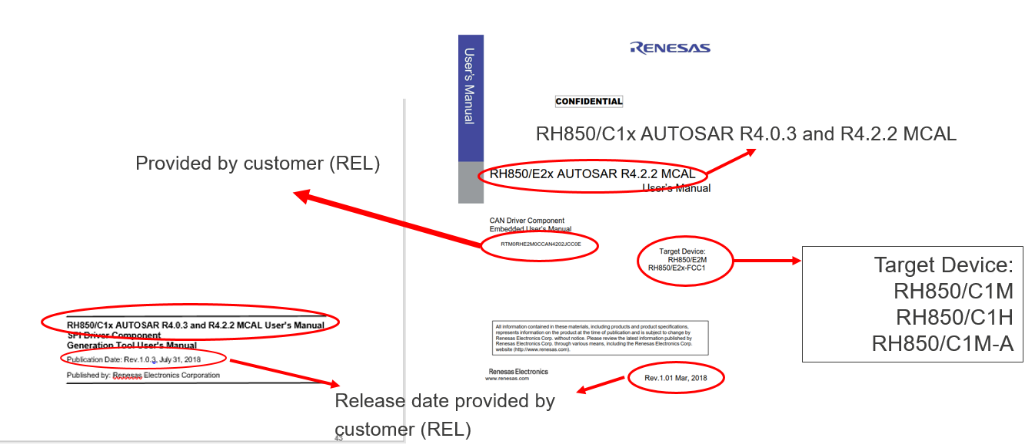
**Release version**: \external\X1X\D1x\modules\<msn>\user\_manual\ \*pdf

**Note:**

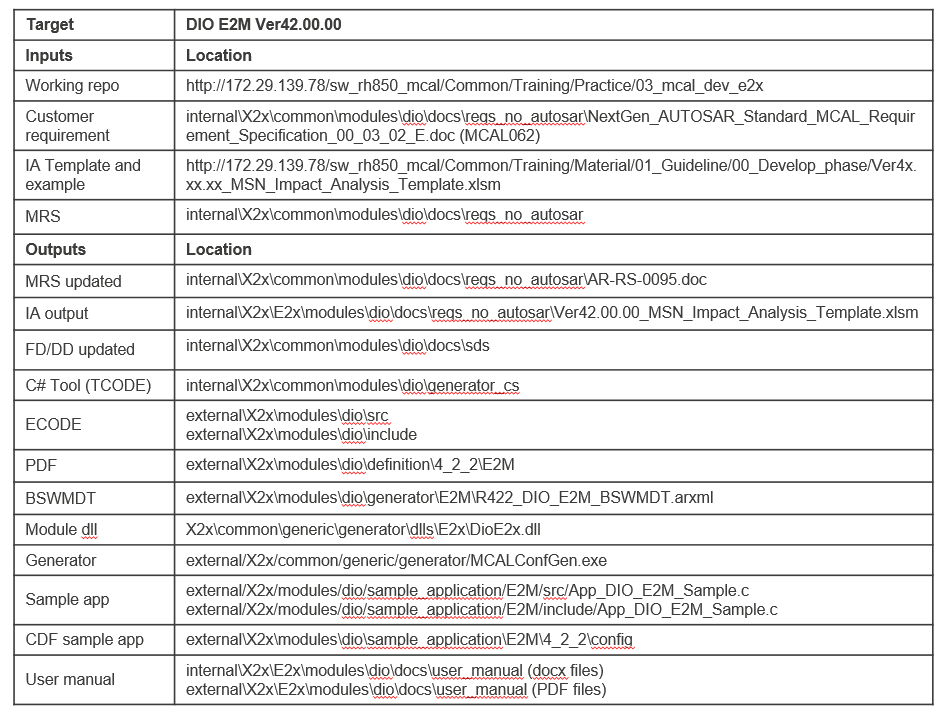
* UM was updated based on Functional Design.
* Release version with \*.pdf format was converted from “Develop version” by Adobe Acrobat Pro installed in a common LabPC



This are some importance information in UM as below:



Below is location of work products, it depends on projects but common location is described as below:

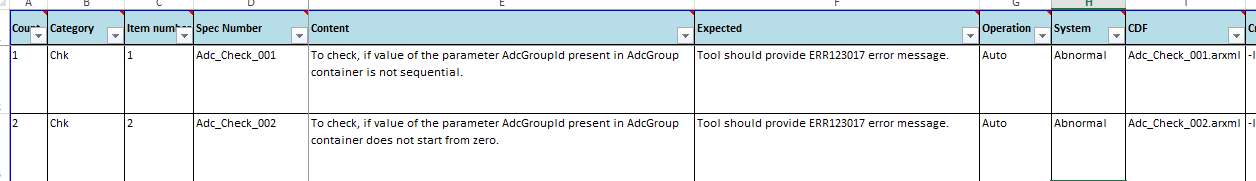


1. Testing

At RVC, there are some types of testing such as Integration Testing (IT), Unit Testing (UT), System Test (ST). Those apply for TCODE and ECODE with named products such as GIT, GUT for TCODE and DIT, DUT for ECODE.

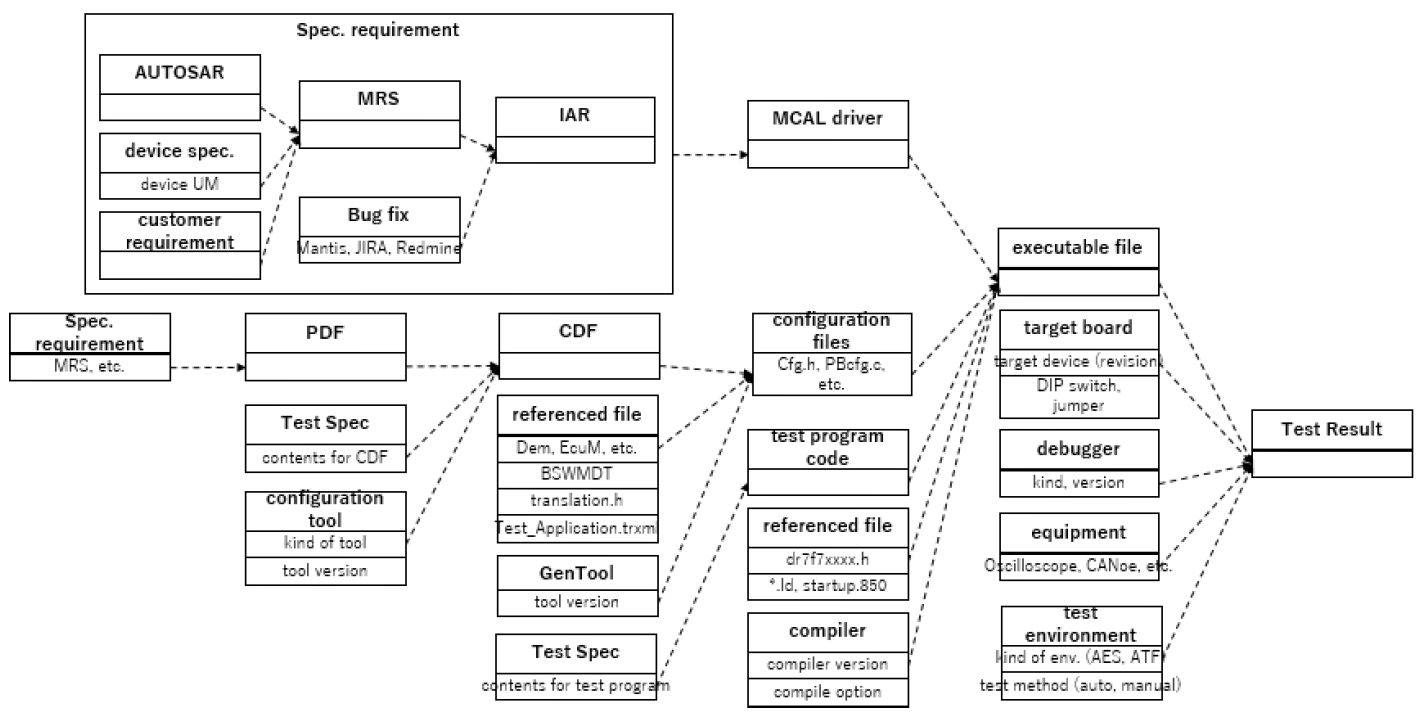
* + GIT stand for Generation Integration Test to confirm the error, warning, information messages are generated when defined conditions are provided and confirm generation tool can handle cautions/limitations or specific requirements correctly. Target of GIT are all cases of error, warning, information should be tested.

An example for testing GIT in module ADC, ERR123017 need testing as description:

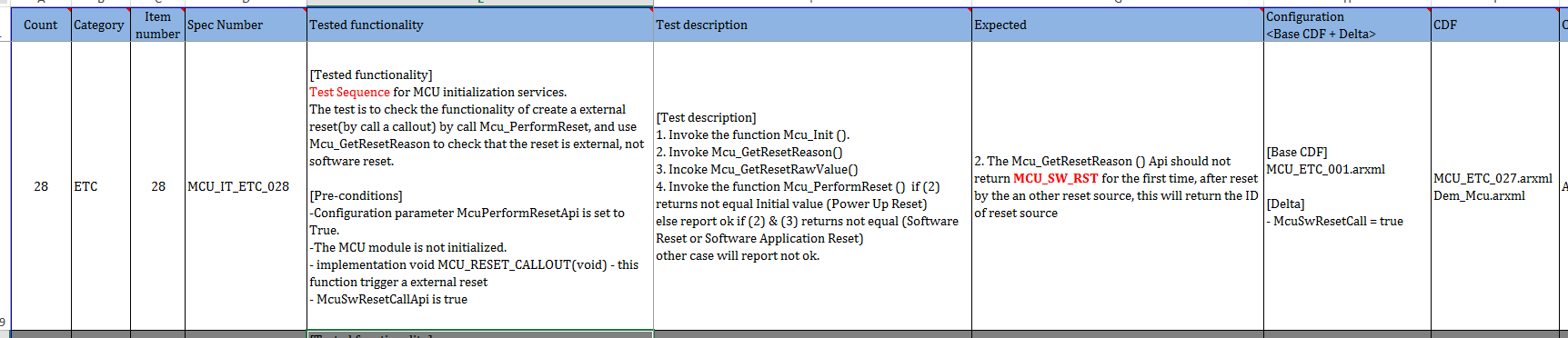


Above is test spec design used to describe content, input and output for each test case in GIT.

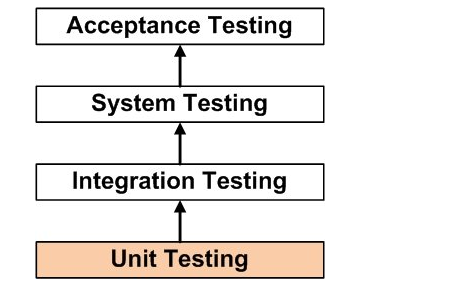
* + DIT stand for Driver Integrated Test to test the functional behavior of the software against functional and non-functional requirements, target of DIT is behavior expectation should be generated. In DIT, board can be required to verify result.



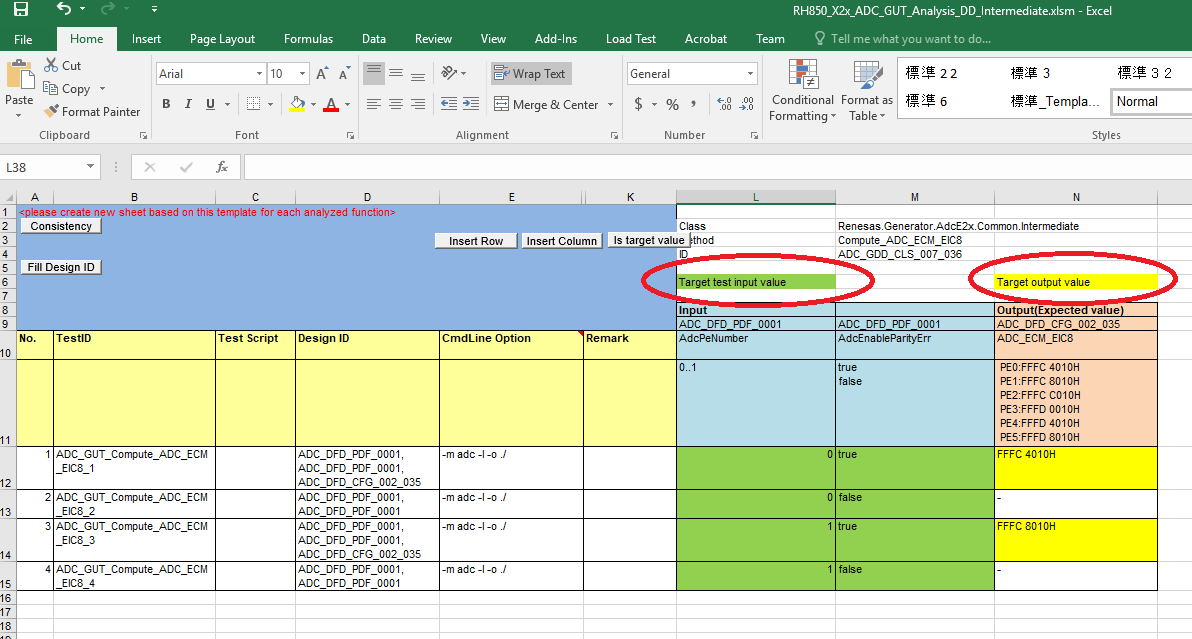
An example test case DIT test spec design of MCU module as below:



* + GUT stand for Generation Unit Test to test individual units/ components of Generation Tool, it is performed by using the White Box Testing method while Integrate Test (IT) using Black Box Testing. Unit Testing is the first level of software testing and is performed prior to Integration Testing.

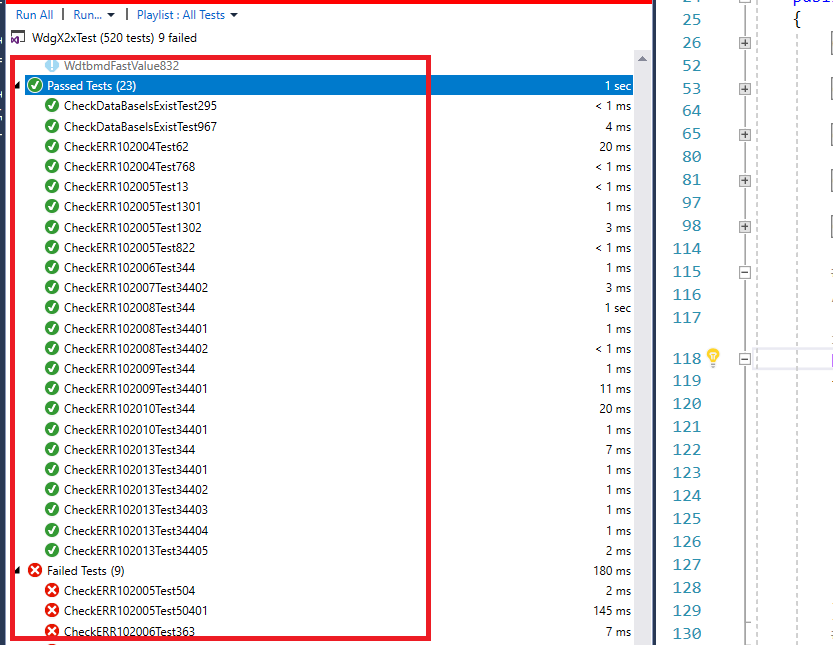


An example GUT test spec design of ADC module as below:



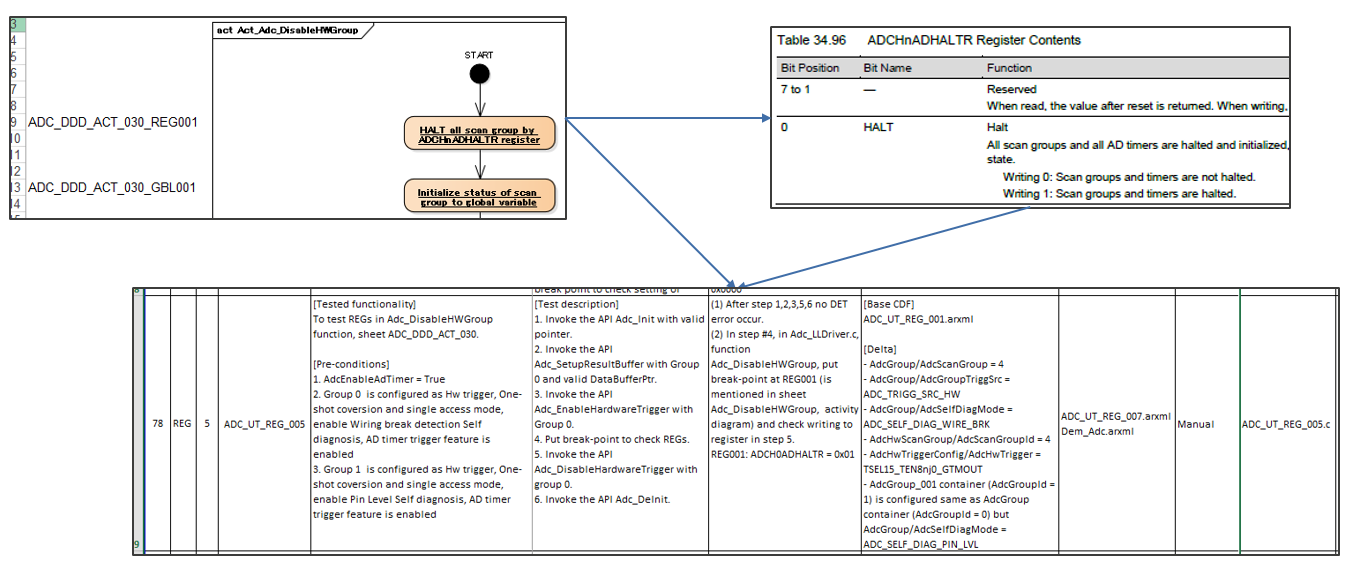
GUT test for method **Compute\_ADC\_ECM\_EIC8()** in Intermediate class. Input and Output should be provided in test spec design.

From test spec design, results are collected by IDE tool after run tests.

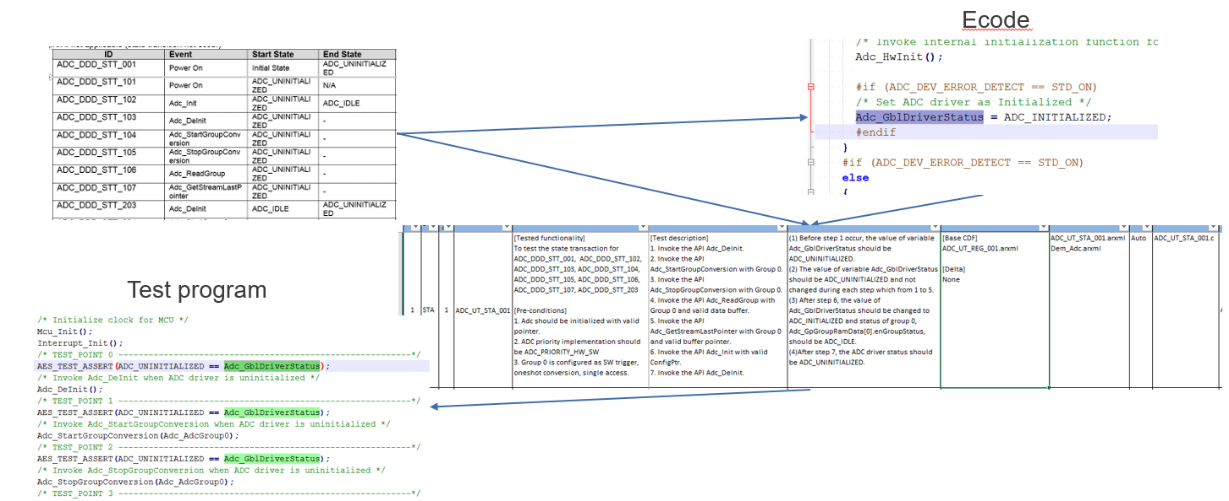


* + DUT stand for Driver Unit Test to test software implementation around the register access, examine register read/write and test functional behavior of the MCAL software with State Transition Testing, target is access and expected read/write value should be correct.

An example DUT for ADC module, based on inputs such as Activity Diagram and HW Manual



An another example:



* + System Testing

Combination testing of MCAL module units in a complex realistic end system defined by a customer to ensure that the behavior of MCAL module on Multi-Core environment is correct.

1. Peer Review

The reviews to be conducted during software development are classified into three types: **peer review, design review, and release review.**

**The peer review** is intended to find defects in work product and improve the quality of the work product through the review by members of the project and experts in the corresponding technological field necessary for review from the department in charge of the development.

**The design review** is intended **to determine the completion** of each process (\*) in the Design Phase through the review by members of the project together with interested persons from the department in charge of the development and the Quality Department. However, in development that does not require software qualification, participation of Quality Department is not necessary.

**The release review** is intended to determine whether the department in charge of the development shall release alpha product or beta product planned in the Project Plan to customers. However, MP release is excluded. For the definition of the release, refer to **"Operation Standard for Software Release"**

Example:

Review file is output of peer review phase in software development at RVC.

