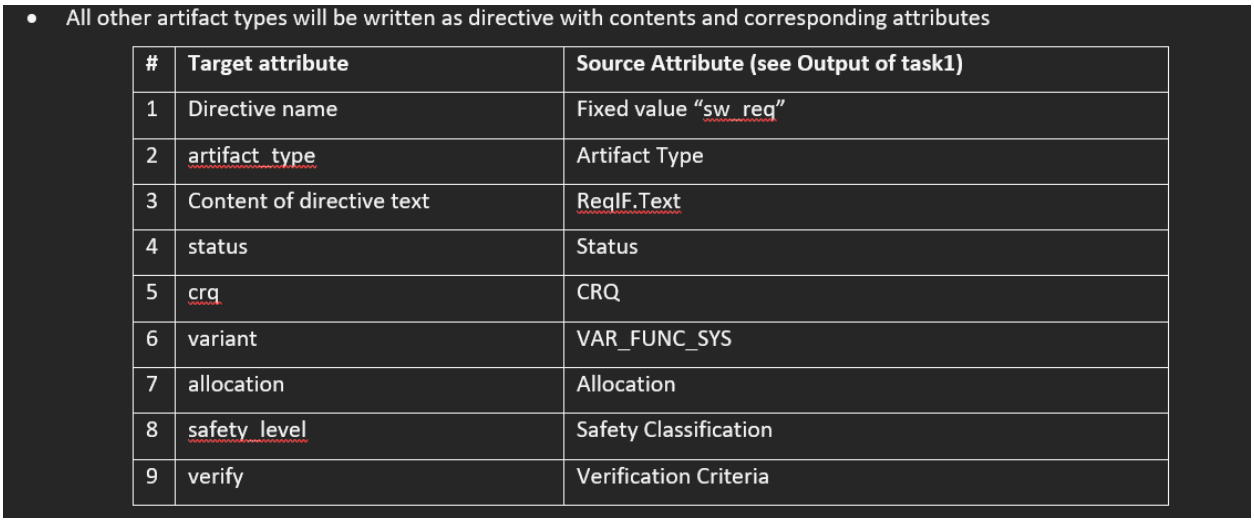
Overall requirement:

Develop a generic mapper and writer using configuration file to parse the ReqIf file, write to RST file and push into GitHub.

We have one input ReqIf file, we can be easy to change the format and content of the RST output file by modify the configuration file without modify source code.

Last round, the mappings between JSON and RST file are fixed and provided by organizers.

- Module name will be written as RST file Heading
- Heading requirement (Artifact Type is Heading) will be written as RST sub-heading
    - Content is ReqIF.Text attribute without formatting
- Information (Artifact Type is Information) will be written as information
    - Content is ReqIF.Text attribute without formatting
- All other artifact types will be written as directive with contents and corresponding attributes



- All other artifact types will be written as directive with contents and corresponding attributes

#	Target attribute	Source Attribute (see Output of task1)
1	Directive name	Fixed value "sw_req"
2	artifact_type	Artifact Type
3	Content of directive text	ReqIF.Text
4	status	Status
5	crq	CRQ
6	variant	VAR_FUNC_SYS
7	allocation	Allocation
8	safety_level	Safety Classification
9	verify	Verification Criteria

Figure 1. Fixed and provided mapping configuration from Round 2.

In this round, you must allow users to configure their own mappings.

Task1: Allow users to configure the way to mapping the attribute name and its value

It will support below cases:

1. Array mapping

   Example: array of the artifacts

2. Attribute name and the same value
   Example:
   Attribute name: CRQ (source) <> crq (target)
   Attribute value: crq target value will be the same as CRQ source value
3. Attribute name and customize value
   Example:
   Attribute name: Status (source) <> status (target)
   Attribute value:
   NEW/CHANGED <> New/Changed
   APPROVED <> Approved
   DELETED <> Deleted
   Default is empty

Input: ReqIf file (same as previous round) and configuration file (can be use single configuration file for task 1 and task 2)

Output:

JSON file with mapped content

Requirement:

Write a java/python command line application to parse the input and write to JSON file with configured file.

Guideline to configure configuration file.

Task 2: Allow users to configure the way to write the requirements.

For example, the module name is written as heading. We can change the module type or module ID will be written as heading.

Similarly, the heading requirement is written as sub-heading, we can config to write it as sub-heading or information or requirement. It will be the same for information and other requirements.

Allow users to configure the way to write the requirement attribute.

One attribute can be written as below format:

1. Attribute value text
2. Sub-directive (always write at the end of requirement, after attributes and content)
3. Html content (convert html to RST text)

Input: JSON file of Task 1

Output: RST file (similar previous round)

Requirement:

Write a java/python command line application to parse the input and write to JSON file with configured file.

Guideline to configure configuration file.

***Task 3*: Automatically upload the RST file from *Task 2* to GitHub.**

Now you must implement a feature which uploads the RST file (output from ***Task 2***) to a particular Github branch (provided by organizers). It can overwrite if the RST file existed.

Input: RST file of task 2

Output: File got pushed to GitHub branch

Requirement:

Write a java/python command line application push the RST file to GitHub and overwrite if the file is existed.

Evaluation

- Task 1: 40
  - Array mapping: 10
  - Attribute name and the same value: 10
  - Attribute name and customize value: 10
  - Easy to configure file: 5
  - Clean code: 5
- Task 2: 40
  - Heading, Information, Directive: 5
  - Attribute value text: 10
  - Sub-directive: 10
  - Html (convert html to RST text): 10
  - Clean code: 5
- Task 3: 20
  - Push to GitHub: 10
  - Overwrite: 5
  - Clean code: 5