**Task 1: Transform the ReqIf (Requirements Interchange Format) file to json format**

Given a Reqif file in xml format, this file saves information about software requirements (**629011\_ECU\_Original\_Requirement.png**). Each requirement consists of attributes such as: Identifier, Attribute type, Status, Description, Name, ……

The main sections of ReqIf file are (see the schema in [link](http://www.omg.org/spec/ReqIF/20110401/reqif.xsd), hints: can use JAX-B / any 3rd party library in java/python to parse XML file):

* THE-HEADER contains some information about the tool, the export and the ReqIf-Format
* CORE-CONTENT defines the content of a specification.

It is structured:

* DATATYPES to specify the types which are used in the document.
* SPEC-TYPES to specify the objects of a document and the document itself
* SPEC-OBJECTS to specify the content of the object
* SPECIFICATIONS to specify the document information (module) and the hierarchy of the objects.
* Each requirement is defined in the sub section SPEC-OBJECT of SPEC-OBJECTS. Each requirement has a unique IDENTIFIER. A requirement is defined by a TYPE and a VALUES. The TYPE references to the section SPEC-TYPES where the types are defined (e.g.: Heading, Information, MO\_NON\_FUNC\_REQ, MO\_FUNC\_REQ …) and a list of attributes. The section VALUES contains the values of attributes.
* Attributes are specified in sub section SPEC-OBJECT-TYPE of SPEC-TYPES, the name is defined in LONG-NAME. Each attribute consists of DEFINITION and THE-VALUE. The DEFINITION references to section DATATYPES. THE-VALUE contains the value of attribute.
* The hierarchy of the objects is defined in section SPEC-HIERARCHY.

**Requirement:** Write a command line program by java or python, the program allows to transform requirements from ReqIf file to JSON file and preserve the values of attributes and object hierarchy.

**Input**: Reqif file as sample attachment (**Requirements.reqif** file)

**Output**: JSON file as sample attachment (**Json\_Output\_Sample.json**) and program source code.

In JSON file need to include below information:

1. Requirement file: Module name (module in specification) and its type
2. All requirements (artifacts) in the same order (with or without hierarchy)
3. Mandatory attributes
   1. Artifact Type (Heading, Information, MO\_FUNC\_REQ, MO\_NON\_FUNC\_REQ)
   2. ReqIF.Text
   3. Status (enumerations)
   4. CRQ
   5. Allocation
   6. VAR\_FUNC\_SYS
   7. Safety Classification (enumerations)
   8. Verification Criteria

**Task 2: Write RST (reStructuredText) file**

After parsing the ReqIf file, we will use the output to transform and write it to RST file (Sphinx-needs).

The requirement will be mapped as below rules:

* 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

|  |  |  |
| --- | --- | --- |
| **#** | **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 |

**Requirement:** Write a command line program by java or python, the program allows to transform JSON file of task 1 to RST file (attached file)

**Input:** the JSON of task 1

**Output:** the RST file (see **ECU\_Requirement.rst**) with above mapping rules and program source code