|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| DISTRIBUTION : | Firm | To | Ref | Copies | 1st page | e-mail |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**How DETAILED DESIGN generation works**

OBJECT

This document describes how to install and configure Detailed Design generator.

SUMMARY

/

CONCLUSION

/

|  |  |  |
| --- | --- | --- |
| STATUS: | For comments | ✓ |
|  | For application |  |

|  |  |  |
| --- | --- | --- |
| Established by | Reviewed by | Approved by |
| Name: Ardeleanu Lucian  Date: 02.11.2021  Visa: | Name:  Date:  Visa: | Name:  Date:  Visa: |

The present document contains **12 pages**, including the flyleaf and the appendices.

# EVOLUTION OF THE DOCUMENT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Issue** | **Date** | **Author** | **Rev.** | **Motive and nature of the modifications** |
| ?????? | 02.11.2021 | Ardeleanu Lucian | 1.1 | First Edition |
| ?????? | 22.11.2021 | Ardeleanu Lucian | 1.2 | Added Details about script |
| ?????? | 03.02.2022 | Ardeleanu Lucian | 1.3 | Modified info about how to run script, added info about yaml config file |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**CONTENTS**

EVOLUTION OF THE DOCUMENT 2

1. RELEVANT DOCUMENTS 4

2. GLOSSARY OF TERMS / EXPRESSIONS 4

3. PREREQUISITES 4

3.1. PTC Source Integrity 4

3.2. DOORS Standalone Application 4

3.3. Reqtify 2018 Standalone Application 4

3.4. Python ( last version ) – NOT MANDATORY, OPTIONAL 4

3.4.1. USED LIBRARIES AND DEPENDENCIES: 4

4. HOW REPORTS ARE GENERATED 5

4.1. Resyncronize Entire Project 5

4.2. Mount S Drive and Remove Read Only 5

4.3. Generate Reqtify Reports 5

# RELEVANT DOCUMENTS

|  |  |  |
| --- | --- | --- |
| **N°** | **Title** | **Reference** |
|  |  |  |

# GLOSSARY OF TERMS / EXPRESSIONS

|  |  |
| --- | --- |
| ***Term*** | ***Definition*** |
| ***BAT*** | *Batch File ( Windows batch file extension )* |
|  |  |
|  |  |

# PREREQUISITES

In order for this tool to work, it is absolutely necessary to map the S: drive.

**All prerequisites used by the Detailed Design generation tool have been installed in the Tools folder.It is not necessary to install any tool.**

If, however, it is necessary to install certain tools, the tools used, in particular, by the generator are Doxygen and its Graphviz plugin.

# HOW REPORTS ARE GENERATED

The tool needed to generate reports is called Detail\_Design\_Generator.exe. This tool must have beside it Tools folder and config file yaml ( Detail\_Design\_Generator\_Config\_file.yaml ).

To generate reports:

1. You need to be in Executables folder. You must find there one folder called Tools and 2 files, one called Detail\_Design\_Generator\_Config\_File.yaml witch will be referred as config file and script file: Detail\_Design\_Generator.exe witch is executable script file.
2. Modify ( using a text editor, ex: Notepad++ ) config file accord your paths and your configuration (**Detail\_Design\_Generator\_Config\_File.yaml** file )
3. Run .exe file: **Detail\_Design\_Generator.exe**

This tool can also work in S: drive. So, if you need to run this tool from here, modify only component name in YAML config file.

For a correct parsing of the source code given at the input, it is necessary that the description before the function respects a certain "template". For more examples of descriptions of these functions, please refer to the source code files of the modules written as an example: First\_Example\_Module and Second\_Example\_Module.

An example of a correct description is the following:

/\*\*

\* **\brief**

\* An example of description for local function. Belongs to First\_Example\_Module.

\* **\param**

\* another\_variable This is a uint8 variable.

\* **\exception**

\* This function has no exceptions.

\* **\pre**

\* This function has no preconditions.

\* **\post**

\* This function has no postconditions.

\* **\return**

\* This function returns a variable.

\*\*/

## How the Detailed Design script works

This tool can work in two major operating modes, namely:

1. When it receives a software component containing a file with the extension "c\_template" at the input, it will ignore the rest of the source code files (.c and .h files in all subfolders of the Implementation folder) and will only analyze the c\_template file. The purpose of these c\_template files will be described below.

2. If no files containing the extension "c\_template" were found in the given component, all the files of the respective module containing the extension .c or .h will be analyzed.

Thus, the script will create a temporary directory called Temp, will copy the source code files within this directory, and the Doxygen tool will be called in this directory to analyze the respective files.

Of all the output generated by Doxygen, only the generated man3 files are used in this script.

After the man3 files are generated by doxygen, these files will be taken over by the tool called mandoc necessary to "read" them. This tool aims to convert them from their man3 form to an easier to read form.

After their conversion with the help of the mandoc, the content of the man3 files will be processed and transformed into different dictionaries that will contain the respective data types.

In case of processing the functions, a specific tool called Moritz (a doxygen plugin) will be called in execution, which aims to generate working graphs of the respective functions. After generating the respective graphs, each graph will be inserted under the table of each parsed function.

WARNING! NOT ALL FUNCTIONS CAN BE PERFORMED BY MORITZ!

When developing the Detailed Design tool, it was noticed that not all functions were correctly parsed by Moritz.

At the end, all the data will be written in the respective tables through the python-docx library and the detailed Design file will be created in the same path where the template is located.

## How to write c\_template files

These c\_template extension files will be searched in the PATH: **S:\Components\Application\Component\_Name\Design.** If they were not found in that PATH, the script will analyze the source code files from the software component implementation folder. ( **S:\Components\Application\Component\_Name\Implementation ).**

These files are intended to generate a detailed design report on a module if there are no source code files on that module or to be written. An example of a c\_template file can be seen in the software component: First\_Example\_Module.

Thus, these files will contain a code similar to a pseudocode, BUT WHICH FOLLOWS THE RULES OF THE C PROGRAMMING LANGUAGE.

/\*\*

\* **\brief**

\* An example of description for local function. Belongs to First\_Example\_Module.

\* **\param**

\* u8FirstVariableExample This is a uint8 variable.

\* **\exception**

\* This function has no exceptions.

\* **\pre**

\* This function has no preconditions.

\* **\post**

\* This function has no postconditions.

\* **\return**

\* This function has no return.

\*\*/

void First\_Example\_Module\_service\_c\_template **(**uint8 variable**)**

**{**

/\* Function code here \*/

/\* Assign to parameter\_1 upper defined value \*/

Variable **=** value\_from\_defined\_macro;

/\* Check values of parameter\_1 \*/

**if** **(** variable\_condition **)**

**{**

/\* Increment it \*/

increment\_variable**;**

**}**

**else**

**{**

/\* Decrement it \*/

decrement\_variable**;**

**}**

**}**

As can be seen, the structure of a source code file c is respected, although it is not compilable. If this structure is not followed, Moritz may not generate graphs on those functions.

The Detailed Design Generator tool will parse each type of date in a certain function in the script, so only the functions needed to parse a certain type in the script arguments, the execution time being in this case very small.