**Project Requirements**

**R.A.D.U - REQUIREMENTS AND DESIGN UTILITY**

**NTT Data Romania S.A.**

**19-21, Constanta Street,**

**400158 Cluj Napoca**

**2019**

**Table of Contents**

[1 Introduction 3](#_Toc14601208)

[2 Requirements 4](#_Toc14601209)

[2.1 XML Requirements Object members 4](#_Toc14601210)

[2.1.1 NeedsCoverage and ProvidesCoverage members: 4](#_Toc14601211)

[2.1.2 Description Member 5](#_Toc14601212)

[2.1.3 Status Member 5](#_Toc14601213)

[2.1.4 CreatedBy Member 6](#_Toc14601214)

[2.1.5 Functional safety relevant 6](#_Toc14601215)

[2.1.6 Change Request Reference 6](#_Toc14601216)

[2.1.7 Review ID 7](#_Toc14601217)

[2.1.8 Type 7](#_Toc14601218)

[2.1.9 Chapter 8](#_Toc14601219)

[2.1.10 Hw/Platform 8](#_Toc14601220)

[2.1.11 Function domain 9](#_Toc14601221)

[2.1.12 Tested at 9](#_Toc14601222)

[2.1.13 Version 10](#_Toc14601223)

[2.2 XML Document settings area. 11](#_Toc14601224)

[2.2.1 Document area members 11](#_Toc14601225)

[2.3 Start Tab 12](#_Toc14601226)

[2.4 Database Tab 13](#_Toc14601227)

[2.4.1 Default columns to display 13](#_Toc14601228)

[2.4.2 Filter functionality 13](#_Toc14601229)

[2.4.3 Save Settings for “filter option” 13](#_Toc14601230)

[2.4.4 Search functionality 13](#_Toc14601231)

[2.4.5 MenuStrip 13](#_Toc14601232)

[2.5 Coverage Tab 16](#_Toc14601233)

[2.5.1 Columns for the dataGrid 16](#_Toc14601234)

[2.5.2 Functionality 16](#_Toc14601235)

[2.6 Statistics Tab 17](#_Toc14601236)

[2.7 Settings Tab 18](#_Toc14601237)

[3 Functionality Requirements 19](#_Toc14601238)

[3.1 Requirements Version 19](#_Toc14601239)

[3.2 Compare between Releases 19](#_Toc14601240)

[3.3 Unsaved requirements 19](#_Toc14601241)

[3.4 Baseline 19](#_Toc14601242)

[3.5 Multiline 19](#_Toc14601243)

[3.6 View Baseline changes 19](#_Toc14601244)

[3.7 Text Style 19](#_Toc14601245)

# Introduction

The purpose of the Requirements Tool Project document is to outline the requirements for the Summer Students Project.

This document contains all of the software requirement specifics. It contains a general description of the types of users who will be using our application, how it is going to work, and what technologies we are using to make it work.

It will also outline and describe each component of the project.

# Requirements

## XML Requirements Object members

### NeedsCoverage and ProvidesCoverage members:

Description:

**User Requirements**

<id> REQ1

<needscoverage> REQ2

<providescoverage> N/A

**System Requirements**

<id> REQ2

<needscoverage> REQ3

<providescoverage> REQ1

**Software Requirements**

<id> REQ3

<needscoverage> src,tst

<providescoverage> REQ2

“src,tst” means that the REQ3 shall be implemented in the code (src) and also tested (tst).

The c# application will search in all code, “.c” and “.h” files, to check if the requirement REQ3 is implemented and tested.

When the developer implements a line of code, is needed to add a comment about the requirement that is implemented. Please check the following example:



**[Req001]**: Default Value for needscoverage shall be: “To be linked”.

**[Req002]**: Default Value for providesciverage shall be: “To be linked”.

**[Req003]**: Providescoverage member needs to be able to open (go to) the requirement from which is derived via hyperlink. ( If the requirement is from the same document, otherwise a message will be popped out “Do you want to open “document x”?)

**[Req004]**: Xml member name:

<needscoverage> and <providescoverage>

### Description Member

Description:

This field is stating the description of the Description of the requirement.

Xml member name:

**[Req005]:** <description>

Possible values:

N/A – (String / text)

Default value:

**[Req006]: “**Please enter the description of the requirement”

Note:

Description shall have the functionality to display pictures.

Format of the picture text:

* **[Req007]:** The pictures shall be saved as “.png” format.
* **[Req100]: The application shall be able to insert pictures in the “Description” Column.**

**Insert the picture under the following format.**

“<Description>

Requirement Description Text.

**Picture1.png**

Requirement Description Text.

</Description>”

The characters that are describing the picture shall be bolded.

* **[Req101]:**

The application shall be able to display the picture when the user clicks over the “Description” cell.

The picture shall be displayed in a new pop up together with the description text.

* **[Req102]:**

If the description contains multiple pictures, the application shall be able to display them all.

Location of the pictures in the project folder tree:

* **[Req008]:** “requirements\Images”

How to version the pictures?

* The pictures will be stored in a folder named “Images” located in the same place as the “.xml” files. The versioning of the pictures is performed via the versioning tool used.

### Status Member

Description:

When you start working on some already existing requirements, you should mark them as “In work “ and check in the xml into version manager in order for the team to be aware that those requirements will be changed.

If you write new requirement, the state should be “Draft” until are finalized.

When you finalize update on requirements, all changed requirements should be set in “Ready for review” state.

After review is done with no findings, all requirements will be switched to “Accepted State”

Description:

* This member is describing the status of the current requirement.
* During the project, a requirement is passing through a chain of events. In the first state, the requirement is defined as a draft. In this state the requirement engineer is adding information about the requirement and at the end when the requirement is fully defined, the “Status” will become “Ready for Review”.
* In this moment the requirement can be reviewed by other person.
* After the review, if the requirement can be implemented in the project, the “Status” will become “Accepted by the project”.

Xml member name:

<status>

[**Req075]**

Possible values:

* Accepted by project
* Ready for review
* Discarded by project
* In work
* Draft

**[Req009]**

Default value:

* Draft

**[Req010]**

### CreatedBy Member

Description:

* This member is describing the author of the requirement. The history of all modifications for the current requirement, can be easily tracked in the Versioning Tool used in the project.

Xml member name:

<CreatedBy>

**[Req011]**

Possible values:

* N/A (string/ text)

**[Req012]**

Default value:

* Author of the requirement.

**[Req013]**

### Functional safety relevant

Description:

Functional Safety is the part of the overall safety of a system or piece of equipment that depends on the system or equipment operating correctly in response to its inputs, including the safe management of likely operator errors, hardware and software failures and environmental changes.

Xml member name:

* <SafetyRelevant>

**[Req014]**

Possible values:

* N/A
* QM
* ASIL A
* ASIL B
* ASIL C
* ASIL D
* **[Req015]**

Default value:

* N/A

**[Req016]**

### Change Request Reference

Description:

* Ticket number based on ID from the issue tracking product (JIRA ID in our case). The ticket that contains the requirement.

Xml member name:

<ChangeRequest>

**[Req017]**

Possible values:

* N/A (IE: Jira ticket number – should be check not to have more then 6 digits)

**[Req018]**

Default value:

* “Change Request ID”

**[Req019]**

### Review ID

Description:

* Ticket number based on ID from issue tracking product (JIRA ID in our case). The ticket for the review.

Xml member name:

* <ReviewID>

**[Req020]**

Possible values:

* N/A (IE: Jira ticket number – should be check not to have more then 6 digits)

**[Req021]**

Default value:

* “Review Ticket ID”

**[Req022]**

### Type

Description:

* This member is describing the type of the requirement.

1. Non-Functional Requirement

This type of requirement is describing an action, a parameter, a functionality from the project. Nothing needs to be implemented in the code or tested.

1. Functional requirement

Is the Requirement that needs to be defined, implemented and tested during the life cycle of the project.

Xml member name:

* <RequirementType>

**[Req023]**

Possible values:

* Description
* Technical Requirement
* Project Requirement
* Functional Requirement
* Non-Functional Requirement
* Template

**[Req024]**

Default value:

* Template

**[Req025]**

### Chapter

Description:

Here you should add the name of chapter for specific requirement ( for ex: req1 is part of Chapter “Introduction to Automotive Area” )

Every Requirement needs to be part of a chapter.

In order to generate a documentation based on this XML, every requirement needs to have specified the Chapter in where it will be generated.

Chapter should be a predefine list in the xml. **[Req029]**

Xml member name:

* <Chapter>

**[Req026]**

Possible values:

* “N/A”
* Predefine list

**[Req027]**

Default value:

* First element from the predefined list.

**[Req028]**

### Hw/Platform

Description:

This attribute contain the platform or hw where the requirement is applicable.

This should be used only for multi-platform projects.

HW/Platform should be a predefine list in the xml **[Req029]**

Xml member name:

* <HWPlatform>

**[Req030]**

Possible values:

* N/A
* The user will define the name of the HW/Platform.
* Predefine list

**[Req031]**

Default value:

* First element from the predefined list.

**[Req032]**

### Function domain

Description:

In case that xml file is a system requirement document, this attribute should be available, otherwise not.

This describe the impact of the requirement in all system domains: sw, mechanical design or hw.

Xml member name:

* <Domain>

**[Req033]**

Possible values:

* N/A
* SW
* MD
* HW

**[Req034]**

Default value:

* N/A

### Tested at

Description:

This attribute is filled in by testing team in order to know each requirement on which level of testing will be covered.

Xml member name:

* <TestedAt>

**[Req035]**

Possible values:

* N/A
* SYS.5
* SYS.4
* SWE.6
* SWE.5
* SWE.4
* Dev Test / Review

**[Req036]**

Default value:

* **N/A**

### Version

Description:

Each requirement have a version number that is increased by 0.1 each time the requirement is changed.

This is used in order to have a good traceability process.

Xml member name:

* <Version>

**[Req037]**

Possible values:

* Version+1

**[Req038]**

Default value:

* 0.1

**[Req039]**

## XML Document settings area.

### Document area members

The application shall be able to select the path for the following:

* Source folder.
* Test folder.

**[Req040]**

## Start Tab

1. Start Tab shall include the title of the project. **[Req041]**
2. Start Tab shall include the NTT logo. **[Req042]**

The logo shall be positioned in the top right of the screen. **[Req043]**

1. Start Tab shall include a short description of the project **[Req044]**

“This project is managing the User Requirements, System Requirements, and Software requirements” **[Req045]**

1. The version of the software shall be included. **[Req046]**

The version shall start from 1.0.0, and to be incremented by 0.0.1.

The history of the software versions shall be stored on the server. (“06\_Software\_Version\_History.docx”)

## Database Tab

### Default columns to display

When a document is opened, the following columns should be displayed by default:

* ID
* Description
* Status
* CreatedBy
* NeedsCoverage
* ProvidesCoverage
* Version

**[Req047]**

### Filter functionality

Under the Edit Bar Menu, the application shall have a button called “Filter”.

From here the user can select the Columns to be displayed on the screen. **[Req048]**

### Save Settings for “filter option”

After the user saves the columns to be showed, the settings shall be saved into XML document.

The informations shall be saved in the xml file under the following format:

<document\_settings>

<doc\_settings>

<Baseline>1</Baseline>

<Columns> ‘columns names’ </Columns>

</doc\_settings>

</document\_settings>

**[Req049]**

### Search functionality

Under the Edit Bar Menu, the application shall have a button called “Search”.

From where the user can input a text, and the application shall display the rows containing that text.

**[Req050]**

### MenuStrip

MenuStrip shall have the following buttons:

* File
* Edit
* **[Req051]**

#### File Button

“File” button shall have the following options(buttons):

* Open

The application shall be able to open “.xml” files.

For any other formats, the following error will be displayed:

“Unsupported file format. Please open supported Requirements file.”

**[Req052]**

* Save

The application shall be able to save the file already opened. The content will be saved over the existing file.

**[Req053]**

* SaveAs

The application shall be able to support Save As functionality. To save the contend in a file with a different name from the already opened one.

**[Req054]**

* Publish

The application shall be able to save the xml in a “pdf”, “word” and “excel” format.

The requirements needs to be in a format easy to read by the customer.

The application will generate the document according to the <Chapter> member from the xml file.

The generated document shall include the following:

* + First page
    - NTT Data address
    - Title (the name of the XML file)
    - Date
  + Table of Contents
    - Generated based on the <Chapter> member from the xml file.
  + Summary
    - This part will include an introduction text like:
    - “A product requirements document (PRD) is a [document](https://en.wikipedia.org/wiki/Document) containing all the requirements to a certain product. It is written to allow people to understand *what* a product should do. A PRD should, however, generally avoid anticipating or defining *how* the product will do it in order to later allow interface designers and engineers to use their expertise to provide the optimal solution to the requirements.

This file is generated based on the requirements document “nameofthedocument.xml”.

**[Req055]**

* Close

The application shall be able to Close previously opened document.

Before actual closing the document, a message shall pop-up with the following text:

“Close Document. Are you sure?”. If the User selects “No” the document will not be closed.

**[Req056]**

#### Edit Button

“Edit” button shall have the following options(buttons):

* Add row

The application shall be able to Automatically insert a row with the default values stated in the previous chapters.

The row will be inserted AFTER the user selected row.

For example if the User is selecting row number 87. The new row inserted will be on position 88.

If no row is selected by the user, the following message shall pop-up “No row is selected for adding a new item”

**[Req057]**

* Delete Row

The application shall be able to Delete a selected row.

If no row is selected by the user, the following message shall pop-up “No row is selected for removal.”

**[Req059]**

* Add Column

The application shall be able to insert a new Column.

The new Column inserted will be placed at the end of the tag lists.

**[Req058]**

* Baseline

The application shall be able to Baseline the document previously opened.

**[Req083]**

## Coverage Tab

### Columns for the dataGrid

The following columns shall be present in the “Coverage” view:

* Requirement ID
* Coverage
  + Possible values:
    - Covered
    - Uncovered
* Path
  + The path in where the requirement was found.
* **[Req060]**

### Functionality

The application shall be able to check the coverage of any requirement saved in the XML file.

**[Req061]**

#### MenuStrip

Menu strip shall have the following buttons:

* ~~Settings~~ **~~[Req062]~~**
* Tools **[Req063]**

##### ~~Settings Button~~

~~“Settings” button shall have the following options(buttons):~~

* + ~~Select Source Path~~ **~~[Req064]~~**
  + ~~Select Test Path~~**~~[Req065]~~**

##### Tools Button

“Tools” button shall have the following options(buttons):

* Make Coverage

Functionality:

* + - Search in the code each ID to check if is implemented or not.

**[Req066]**

* Publish Coverage
  + - Save the coverage raport into a Excel file. This file needs to be versioned on the server.**[Req067]**

## Statistics Tab

The application shall be able to generate the following statistics:

1. Percent of the covered vs uncovered requiremens

- display a Pie Chart

**[Req068]**

b) Display “Version Mismatch” requirements.

- A version mismatch requirement is a requirement that is having a version stated in the xml file. And another version in the source.

**[Req069]**

IE

Xml:

Requirement: X

Version: 1.1

Main.c

Requirement: x

Version 1.0

## Settings Tab

The Tab will be named “Project Settings”

**[Req080]**

The Application shall be able to configure the “Chapter” and the “HWPlatform”.(and all other dynamic parameters).

**[Req081]**

* The following buttons shall be implemented:
  + Select Source Path **[Req064]**
  + Select Test Path**[Req065]**

## About Tab

The tab will be named “About”

The application shall be able to store the information about the Project. Version. Developers name etc.

**[Req113]**

Information about the Project shall include also the “User Documentation” via hyperlink to a pdf document stored.

**[Req114]**

# Functional Requirements

## Requirements Version

When a version is changed from a “lower requirement” that covers another “higher requirement, the “higher requirement” shall become “uncovered”.

**[Req072]**

When a new requirement is added, the application shall check before save if any column is TBD. In case is TBD, a pop up warning should appear to ask you to check again. Save will not be done.

**[Req073]**

The Tool should be able to export data based on the active filters.

**[Req074]**

The format shall be PDF, Excel Word.

**[Req075]**

## Compare between Releases

The application shall be able to count, how many requirements are modified between two releases.

**[Req070]**

The application shall be able to export the releases that has been modified between two releases.

**[Req071]**

## Unsaved requirements

When a requirement is modified, the application shall color the box with yellow.

This information shall be present after Close and Reopen the document.

When the Baseline is changed. All the Unsaved (Yellow) requirements shall become Saved (Gray).

**[Req076]**

## Baseline

The Application shall be able to Baseline the Document

**[Req077]**

## Multiline

The application shall be able to display the “Description” to multiple lines.

**[Req078]**

## View Baseline changes

The application shall be able to display the requirements that are changed from the previous Baseline.

**[Req079]**

## Text Style

Text shall be centered for all columns except the “Description” column.

**[Req082]**

The text style for the exported Excel file shall adhere to the requirement [Req082].

**[Req103]**

# Document Header

The requirements document shall have a Header with the following structure:

1. Introduction
   1. Purpose of the Document
   2. Intended Use
2. General description
3. Requirements

**[Req104]**

The header shall be stored in the xml file under <DocumentIntro> tag.

**[Req105]**

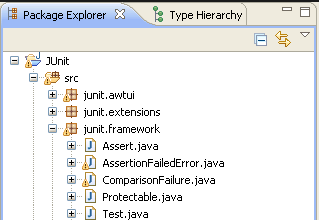
The header shall be displayed in the application right above the DataGridView.

**[Req106]**

# Hierarchy Menu

The application shall be able to display each requirement according to the corresponding chapter on the left of the screen.

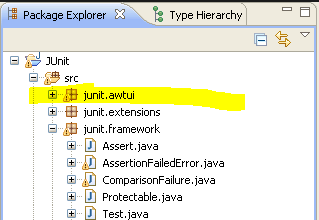
I.E.



**[Req107]**

In the Hierarchy Menu, the Chapters shall have the possibility to minimize and hide all containing requirements.

I.E.



**[Req108]**

When the user clicks on one requirement, the DataGridView shall jump to that requirement.

**[Req109]**

# Right Click Functionality

Right Click must have the following buttons:

* Add Row

**[Req110]**

* Delete Row

**[Req111]**

* Duplicate Row Above

**[Req112]**

* Duplicate Row After

**[Req115]**