**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](#_Toc6078502)

[2 Requirements 4](#_Toc6078503)

[2.1 XML Requirements Object members 5](#_Toc6078504)

[2.1.1 NeedsCoverage and ProvidesCoverage members: 5](#_Toc6078505)

[2.1.2 Description Member 6](#_Toc6078506)

[2.1.3 Status Member 6](#_Toc6078507)

[2.1.4 CreatedBy Member 6](#_Toc6078508)

[2.1.5 Priority Member 6](#_Toc6078509)

[2.1.6 Functional safety relevant 6](#_Toc6078510)

[2.1.7 Change Request Reference 7](#_Toc6078511)

[2.1.8 Review ID 7](#_Toc6078512)

[2.1.9 Type 7](#_Toc6078513)

[2.1.10 Chapter 7](#_Toc6078514)

[2.2 Version 7](#_Toc6078515)

[2.3 XML Document settings area. 7](#_Toc6078516)

[2.3.1 Document area members 7](#_Toc6078517)

[2.4 Start Tab 8](#_Toc6078518)

[2.5 Database Tab 9](#_Toc6078519)

[2.5.1 Default columns to display 9](#_Toc6078520)

[2.5.2 Filter functionality 9](#_Toc6078521)

[2.5.3 Save Settings for “filter option” 9](#_Toc6078522)

[2.5.4 Search functionality 9](#_Toc6078523)

[2.5.5 MenuStrip 9](#_Toc6078524)

[2.6 Coverage Tab 10](#_Toc6078525)

[2.6.1 Columns for the dataGrid 10](#_Toc6078526)

[2.6.2 Functionality 10](#_Toc6078527)

[2.7 Statistics Tab 11](#_Toc6078528)

[3 Functionality Requirements 12](#_Toc6078529)

[3.1 Requirements Version 12](#_Toc6078530)

# 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

| **Requirement ID** | **Description** | **Status** |
| --- | --- | --- |
| 1. | XML “needscoverage” member shall be implemented. Please see 3.1.1 chapter |  |
| 2. | XML “providescoverage” member shall be implemented. Please see 3.1.1 |  |
| 3. | Each requirement from the xml file shall support multiple members defined in chapter 3.1 |  |
| 4. | The xml file shall support a document settings area. Please see 3.2 chapter. |  |
| 5. | The C# Application shall have implemented a tab called “Start”. Please see chapter 3.3 |  |
| 6. | The C# Application shall have implemented a tab called “Database”. Please see chapter 3.4 |  |
| 7. | The C# Application shall have implemented a tab called “Coverage”. Please see chapter 3.5 |  |
| 8. | The C# Application shall have implemented a tab called “Statistics”. Please see chapter 3.6 |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## 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:



Default Value for needscoverage shall be: TBD **(“To be linked”)**

Default Value for providesciverage shall be: TBD**(“To be linked”)**

TDB: 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”?)**

### Description Member

Description:

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

Xml member name:

**<description>**

Possible values:

**N/A**

Default value:

**“Please enter the description of the requirement”**

Note:

Description shall have the functionality to display pictures.

Format of the picture text:

* **The pictures shall be saved as “.png” format.**

Location of the pictures in the project folder tree:

* **“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.

When you finalized 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>**

Possible values:

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

**Default value:**

* In work

### 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>**

Possible values:

* **N/A**

Default value:

* **Author of the requirement.**

### 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>**

Possible values:

* N/A
* QM
* ASIL A
* ASIL B
* ASIL C
* ASIL D

Default value:

* **“Software Safety”**

### 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>**

Possible values:

* **N/A (IE: Jira ticket number)**

Default value:

* **“Change Request ID”**

### 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>**

Possible values:

* **N/A (IE: Jira ticket number)**

Default value:

* **“Review Ticket ID”**

### Type

Description:

* **This member is describing the type of the requirement.**

1. **Description – 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.**
2. **Technical Requirement**

**The purpose of the Technical Requirements Process is to establish the technical requirements of the acquisition. This involves the elicitation of functional and non-functional requirements that consider the deployment life cycle of the products so as to establish a technical requirement baseline.**

1. **Project requirement**

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

Xml member name:

* **<RequirementType>**

Possible values:

* Description
* **Technical Requirement**
* **Project Requirement**

Default value:

* **“Requirement Type”**

### Chapter

Description:

Here you should add the name of chapter for specific requirement ( for ex: req1 is part of Chapter DCM )

**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.**

Xml member name:

* **<Chapter>**

Possible values:

* **N/A**
* **The user will define the name of the Chapters.**

Default value:

* **“Chapter 1”**

### Release

Xml member name:

* **<Release>**

Possible values:

* **N/A**
* **The user will define the name of the Release.**

Default value:

### Hw/Platform

Xml member name:

* **<HWPlatform>**

Possible values:

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

Default value:

* **“HW/Platform”**

## XML Document settings area.

### Document area members

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

* **Source folder.**
* **Test folder.**

## Start Tab

1. Start Tab shall include the title of the project.
2. Start Tab shall include the NTT logo.

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

1. Start Tab shall include a short description of the project

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

1. **The version of the software shall be included.**

**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**

### 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.**

### Save Settings for “filter option”

TBD: 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>**

### 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.**

### MenuStrip

MenuStrip shall have the following buttons:

* File
* Edit

#### 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.”**

* Save

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

* 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.**

* Publish

**The application shall be able to save the xml in a “pdf” 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”.**

* Close
* **The application shall be able to Close previously opened document.**
* **Beforeactual 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.**

#### 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”**

* 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.”**

* **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.**

## 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.**

### Functionality

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

#### MenuStrip

Menu strip shall have the following buttons:

* Settings
* Tools

##### Settings Button

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

* **Select Source Path**
* **Select Test Path**

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

## Statistics Tab

**The application shall be able to generate the following statistics:**

1. **Percent of the covered vs uncovered requiremens**

**- display a Pie Chart**

**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.**

**IE**

**Xml:**

**Requirement: X**

**Version: 1.1**

**Main.c**

**Requirement: x**

**Version 1.0**

# Functionality Requirements

## Requirements Version

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

TDB

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.

Tool should be able to export data like:

How many requirements are based on a filter (filter will be a merge of columns for ex: status :

## Compare between Releases

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

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