IBM Rational DOORS
Requirements Management Framework Add-On

USER MANUAL

IBM Rational DOORS Requirements Management Framework Add-on

User manual

Release 6.1

Before using this information, be sure to read the general information under the "Notices" chapter on page 151.

This edition applies to **VERSION 6.1, IBM Rational DOORS Requirements Management Framework Add-on** and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 2009,2013
US Government Users Restricted Rights—Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Table of Contents

1 OVERVIEW....9

2 INTRODUCTION....10

2.1 DOORS VERSUS IRDRMFAO....10

2.2 RMF GENERIC DATA MODEL....10

2.3 DATABASE AND DOCUMENT APPROACHES....13

3 GETTING STARTED WITH IRDRMFAO....14

3.1 RMF PROJECT INITIALIZATION....14
3.1.1 CREATE A NEW RMF PROJECT....14
3.1.2 MIGRATE AN EXISTING DOORS PROJECT INTO RMF FORMAT....15
3.1.3 RMF PROJECT CONFIGURATION....16
3.1.3.1 PUID....17
3.1.3.2 WORD....20
3.1.3.3 RCM....21
3.1.3.4 PFM....21
3.1.3.5 DOC....23
3.1.3.6 EXCHANGE....24
3.1.3.7 CHECK....25
3.1.3.8 VISIBILITY....27
3.1.4 PROJECT INDEX INITIALIZATION....30

3.2 RMF MODULE INITIALIZATION....32
3.2.1 WHAT IS A MODULE TYPE AND A TEMPLATE ?....32
3.2.2 CREATE A NEW RMF Module....36
3.2.3 MIGRATE A EXISTING DOORS MODULE INTO RMF FORMAT....38
3.2.4 MIGRATE SEVERAL EXISTING DOORS MODULES INTO RMF FORMAT....39
3.2.5 RMF MODULE CONFIGURATION....41
3.2.5.1 PUID....42
3.2.5.2 Objects....43
3.2.5.3 Styles....44
3.2.5.4 RCM and RCM attributes....46
3.2.5.5 Check....48

3.3 DEFINE THE DEFAULT LINKSET PAIRING....49

4 REQUIREMENT ANALYSIS....52

4.1 CHARACTERIZE REQUIREMENTS....52

4.2 DEFINE ISSUES AND DECISIONS....53

4.3 Identify RISK and key requirements....54
4.3.1 Introduction....54
4.3.2 critical requirements....54
4.3.3 Key requirements....54

4.4 LINK RMF OBJECTS....55
4.4.1 LINK SET SELECTION....55
4.4.2 LINKING RMF OBJECTS....55

4.4.2.1 GENERAL PRINCIPLES....55
4.4.2.2 Requirements satisfY upper level requirements....56
4.4.2.3 REQUIREMENTS / FUNCTIONS ARE ALLOCATED TO CONFIGURATION ITEMS 56
4.4.2.4 LINK RMF OBJECTS WITHIN A SAME MODULE....56
4.4.3 CHECK THAT YOUR PROJECT LINKS CONFORM WITH DATA MODEL....56

4.5 CHECK DATA CONSISTENCY....57
4.5.1 CHECK PROJECT AND MODULE CONSISTENCY TOOL....57
4.5.2 INTEGRITY CHECK....60
4.5.2.1 Integrity check at project level....63
4.5.2.2 Integrity check at module level....64
4.5.2.3 Integrity check report dialog....65
4.5.2.4 Integrity check in triggers....67
4.5.2.5 Predefined integrity rules....68
4.5.2.6 User defined integrity rules....69

5 DEFINING THE IVV SOLUTION....71

5.1 Introduction....71

5.2 IVV OBJECT ATTRIBUTES....71
5.2.1 IE Object type....71
5.2.2 IE PUID....71
5.2.3 IE IVV Type....72
5.2.4 IE IVV Test Method....72
5.2.5 IE IVV APPROVAL LEVEL....72
5.2.6 IE IVV responsible....72
5.2.7 IE IVV SKILLS....72
5.2.8 IE IVV EVENT....72
5.2.9 IE IVV EVENT PROVIDER....73
5.2.10 IE IVV Non Regression....73
5.2.11 IE IVV Test SCOPE....73

5.3 IVV RELATIONSHIPS....73
5.3.1 JUSTIFICATION....73
5.3.2 UNDER ISSUE PROCESS....73
5.3.3 VERIFICATION....73
5.3.4 ALLOCATION....74

5.4 IVV VIEWS....74
5.4.1 STANDARD VIEW....74
5.4.2 IVV ASSOCIATED ISSUES VIEW....74
5.4.3 IVV DOCUMENT VIEW....74
5.4.4 IVV MATRIX VIEW....74
5.4.5 IVV PROCEDURES DEFINITION VIEW....74
5.4.6 IVV PROCEDURES PLANNING VIEW....74
5.4.7 IVV JUSTIFICATION VIEW....74
5.4.8 IVV TEST EQUIPMENT VIEW....75

6 MANAGING YOUR DATA FROM A DOCUMENT POINT OF VIEW....76

6.1 Introduction....76

6.2 The document view....76

6.3 Editing paragraph styles....77
6.3.1 Using IRDRMFAO tool “Manage Object”, “Edit Style” tab....77
6.3.2 The DOORS standard tool “Edit paragraph style attribute”....78

6.4 Exporting DOORS data into a WORD document....78

6.4.1 The standard DOORS WORD exporter....79
6.4.2 The Enhanced Word Exporter (WEXP)....80

7 DATA MODEL CUSTOMIZATION....83

7.1 DESCRIPTION OF A RMF PROJECT STRUCTURE....83

7.2 ADAPTING MODULE ATTRIBUTES....84

7.3 ADAPTING PROJECT PROFILE....85
7.3.1 ADD A NEW OBJECT TYPE....87
7.3.2 ADD A NEW TEMPLATE....88
7.3.3 ADD A NEW MODULE TYPE....89
7.3.4 ADD A NEW RELATIONSHIP....90
7.3.5 CHECK THE MODEL....91
7.3.6 ERROR HANDLING....92

8 IMPORT AND IDENTIFICATION OF RMF OBJECTS....94

8.1 Introduction....94

8.2 RMF Object structure within a module....94
8.2.1 Requirement Type Structures....94
8.2.2 Breakdown Type Structures....94
8.2.3 Issue/Decision Type Structures....95

8.3 HOW TO Import a document into DOORS....95

8.4 HOW TO IDENTIFY RMF OBJECTS: REQUIREMENTS, FUNCTIONS,....96
8.4.1 Introduction....96
8.4.2 Identifying an existing DOORS object....96
8.4.2.1 Identifying a selection of objects....96
8.4.2.2 Applying a MS Word Style....97
8.4.2.3 Parsing the text of the identified object....97
8.4.3 Creating new RMF objects....98

8.5 WORKING IN EDIT SHAREABLE MODE....98

8.6 Defining IRDRMFAO behaviour for New module Types....100

9 COLLECTING METRICS....101

10 MANAGING REQUIREMENT CHANGES....102

10.1 Introduction....102

10.2 Managing changes originating outside doors....102
10.2.1 INTRODUCTION....102
10.2.2 1st step: import of a new version of the source document....103
10.2.3 2nd step: comparison of the two versions of the source document....103
10.2.3.1 GUI and setup....103
10.2.3.2 Pre-processing verifications....106
10.2.3.3 Processing verifications and log window....106
10.2.3.4 Classic algorithm....107
10.2.3.4.1 Use and Parameters....107
10.2.3.4.2 Final result window....110
10.2.3.4.2.1 Standard (i.e. non reverse) comparison....110
10.2.3.4.2.2 Reverse comparison....110
10.2.3.5 Hierarchical algorithm....112

10.2.3.5.1 Use and Parameters....112
10.2.3.5.2 Save confirmation window....116
10.2.4 3rd step: Human check....116
10.2.5 4th step: Transfer analysis....117
10.2.5.1 Graphic User interface....117
10.2.5.1.1 Simple GUI (i.e. collapsed advanced panel)....117
10.2.5.1.2 Advanced GUI i.e. (including the advanced panel)....118
10.2.5.2 Use and Parameters....119
10.2.5.3 Pre-processing verifications and log window....121
10.2.5.4 Processing verifications and log window....123
10.2.6 5th step: optional deletion of the older version of the source document....123

10.3 Managing change within DOORS....124

APPENDIX A. USER REQUIREMENTS MODULE DESCRIPTION FORM 125

APPENDIX B. SYSTEM REQUIREMENTS MODULE DESCRIPTION FORM 130

APPENDIX C. PBS MODULE DESCRIPTION FORM....135

APPENDIX D. ISSUES AND DECISIONS AND JUSTIFICATIONS MODULE DESCRIPTION FORM....138

APPENDIX E. IVV PROCEDURES MODULE DESCRIPTION FORM……141

APPENDIX F. PROJECT PROFILE MODULE DESCRIPTION FORM…..145

APPENDIX G. FREQUENTLY ASKED QUESTIONS (FAQ)....150

NOTICES....151

1 Overview

This document relates to the requirements management add-on **IBM® Rational® DOORS® Requirements Management Framework Add-on**, based upon **IBM® Rational® DOORS®**.

IBM® Rational® DOORS® Requirements Management Framework Add-on is a solution which will save your time starting your project with one of the best possible practices you can handle with DOORS. Originating from Thales company, and previously delivered under the name DOORS T-REK, **IBM® Rational® DOORS® Requirements Management Framework Add-on** is a solution developed for the industry by industrials. It implements a methodology in compliance with EIA632, ISO 15288 and CMMI.

IBM® Rational® DOORS® Requirements Management Framework Add-on can be used by System Engineers, Software Engineers, Hardware…a wide set of disciplines.

IBM® Rational® DOORS® Requirements Management Framework Add-on adds a process, a data model and utilities to DOORS. This user manual describes how to use the data model and how to apply the tool.

In the body of the document the acronym **IRDRMFAO** will frequently be used to designate the product **IBM® Rational® DOORS® Requirements Management Framework Add-on.**

The acronym **RMF** will be used to qualify an operation of the product, or a piece of information managed by the product.

Note: The word "project" used in the following is generic and could designate an industrial project, a contract under negotiation, but it is in general used to designate a DOORS "project", i.e. an item of type "project" in the DOORS database.

2 Introduction

2.1 DOORS VERSUS IRDRMFAO

IRDRMFAO is a solution that implements a methodology in compliance with EIA632, ISO 15288 and CMMI.

To achieve this, **IRDRMFAO** adds a process, a data model and utilities to DOORS. Notice that **IRDRMFAO** doesn't hide DOORS layout, all DOORS commands are still available to users.

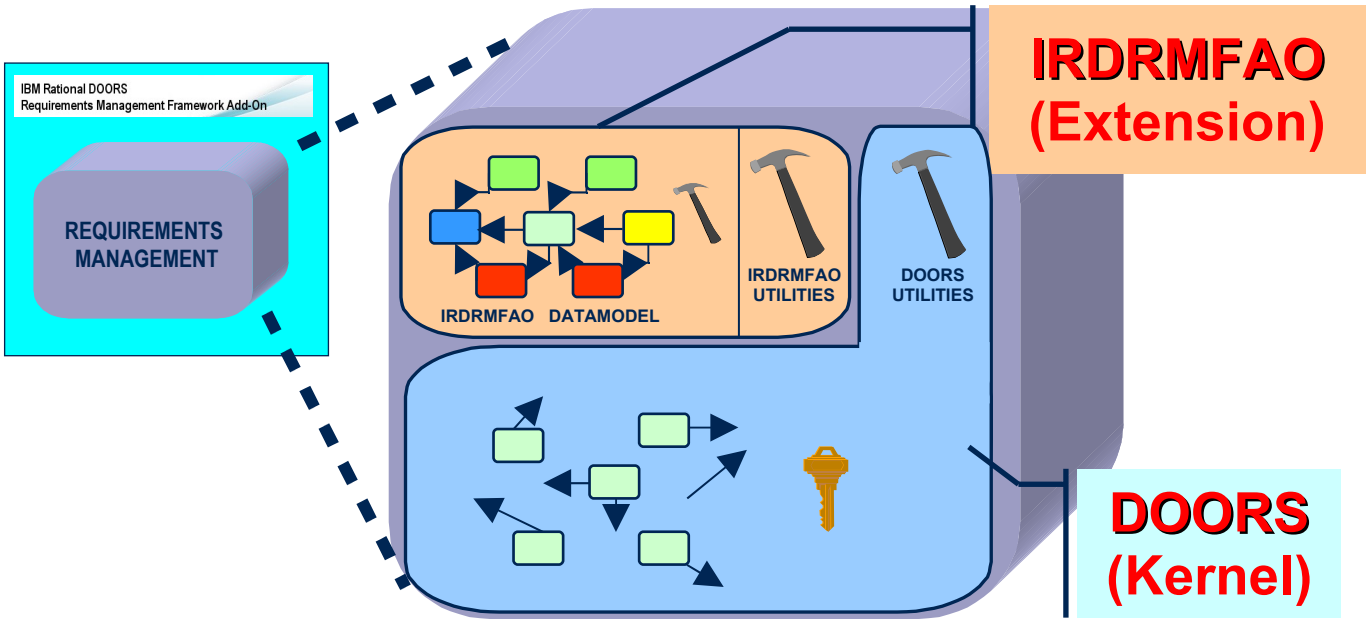


Figure 1: Product Layout

2.2 RMF GENERIC DATA MODEL

IRDRMFAO proposes a generic data model but does not constraint you to use it precisely. In fact, it implements a collection of module types and relationships that you will pick from to build your own project to fit your specific needs. The most generic part of a RMF data model is the list of module types (also named templates) and the possible relationships between them. A data model may also define a project organization (for example all your projects may be based on the same product breakdown structure), and it also may contain predefined documents (i.e. formal modules in the DOORS database), instances or not of some the model templates.

The following figure shows an example of a **RMF** model. It is composed of module types and relationships between them.

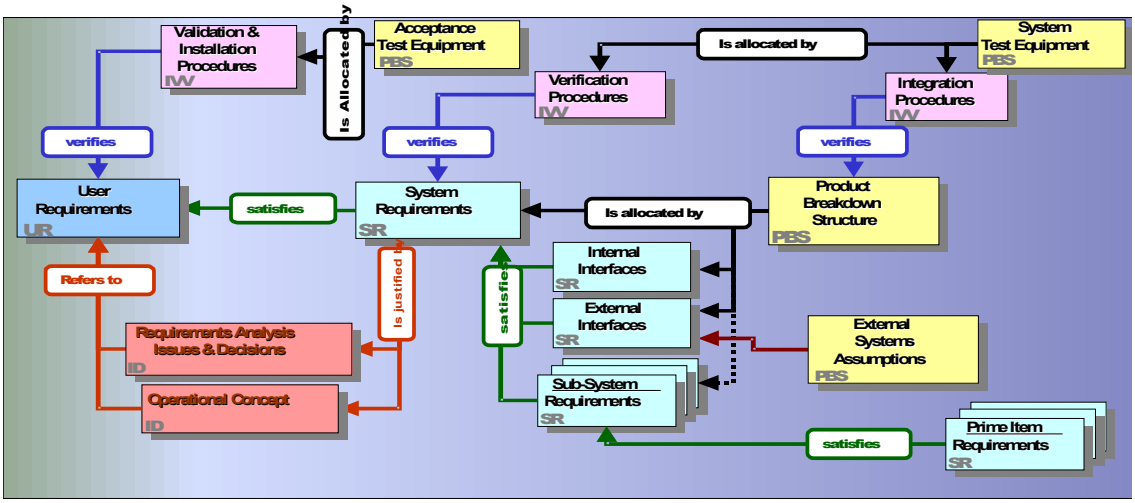


Figure 2 : RMF model example

Each box represents a module type (used to instantiate DOORS formal modules), connected between them by relationships (implemented with DOORS link modules and linksets). Note that in this data model the links are bottom-up oriented. This feature assists impact analysis and traceability, but more importantly it means that links can be added even if the current user does not have write access to the higher level module. All links can of course be browsed in either direction.

- This data model is adaptable to your project. **IRDRMFAO** does not constraint it; you can use some parts of it and if needed, progressively implement it. Eventually, you may modify it entirely. Each module of a certain type is not necessary unique. For example, you can have multiple modules of type "User Requirements" or "System Requirements". In short, the same data model can be used to fit to your project (small and large ones).

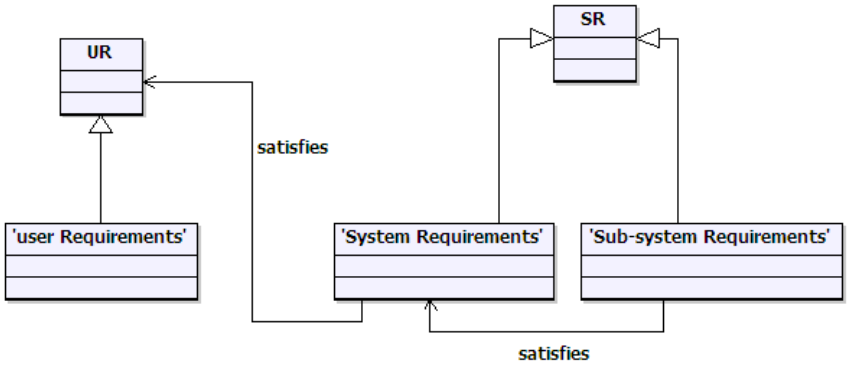
In a lot of cases, different module types may contain identical information from a logical point of view. For example a "System Requirements" module type and a "Sub-System Requirements" module type contains requirements, with similar attributes. To simplify the construction of the data model, these two module types will be implemented with the same template, "SR".

Frequently in the document we don't make any difference between the "Template" concept and the "Module Type" concept. A "Module Type" can be seeing as a specific usage of a template. A "RMF Module" is an instance of a "Module Type", and the content of the "Module Type" is derived from the parent "Template".

Briefly, each activity supported by the Requirements Management process deals with a part of the data model, and uses different kinds of modules types (or templates) and relationships:

- System requirements analysis: UR, SR, IDJ modules and "satisfies", "is justified by" and "refers to" links,
- Design analysis: SR, PBS, IDJ modules and "is allocated by", "is justified by" and "refers to" links,
- Validation tests: IVV, UR modules and "verifies" link,
- Verification and integration tests: IVV, SR modules and "verifies" link.

Example of a simple model described with an UML diagram:



For more detail on the modules, consult the appendix.

2.3 DATABASE AND DOCUMENT APPROACHES

IRDRMFAO allows the user to swap instantly between document or database visualization of the same formal module.

The document approach is used to:

- get a global and linear visualization,
- display structured information (Section, chapters, paragraph,…),
- create a snapshot or baseline in a book format (export) for communicate with customers, contractors,…

The database approach is used to:

- analyze information and characterize data with attributes,
- concentrate the relevant information (filters)
- display traceability matrices.

The instantaneous display swapping between the two approaches is performed through the predefined views in **RMF** modules.

3 Getting started with IRDRMFAO

The steps necessary to create a **RMF** project are shown below:

- Project initialization (Create a **RMF** structure)
- Define and fill your own project structure

3.1 RMF PROJECT INITIALIZATION

It is assumed that:

- DOORS and **IRDRMFAO** have already been installed and you have valid licenses
- A DOORS database has been created with users, groups and access rights defined
- The reader is familiar with DOORS.

3.1.1 CREATE A NEW RMF PROJECT

To create a new **RMF** project:

- Run DOORS,
- You must have the **power to create new projects** (see your user profile) to be able to create a RMF project,
- In the project manager window (also called "Database window"), select a location with the DOORS Explorer (left part of the "Database window"). You should see the root of database called "DOORS Database", otherwise select the menu "View ->Database view",
- From the project manager window, call the menu "**RMF** ->Create/Tag a **RMF** project",
- To create a new **RMF** project, give a name and a description to new project. Both fields are mandatory. Check if your company has adopted naming convention. Note that project name and description can be modified later,
- Select a data model in the list (a company can create several customized data model). By default select the Generic data model,
- The option "Launch the project configuration tool" allows to directly follow on a other tool described in the next paragraph. We suggest to unset it for this first try,
- Click on the "Create" button, if successful, an acknowledgement window is then displayed.

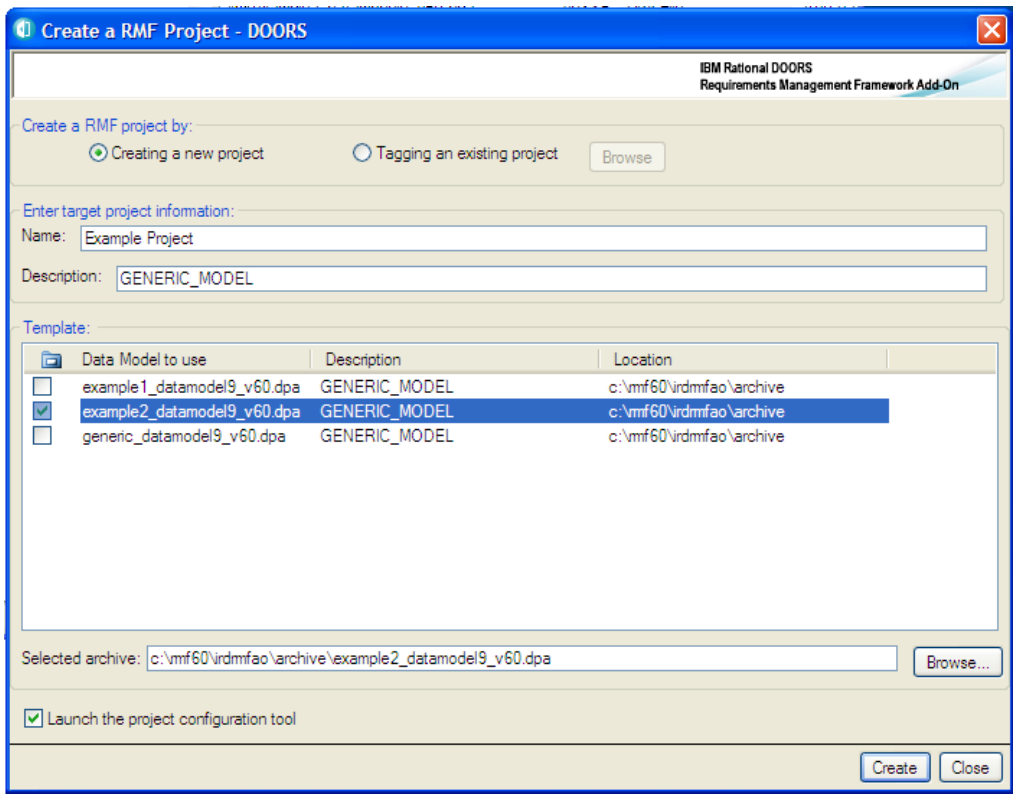


Figure 3 : Create a new RMF project dialog box

The project is created with some predefined content:

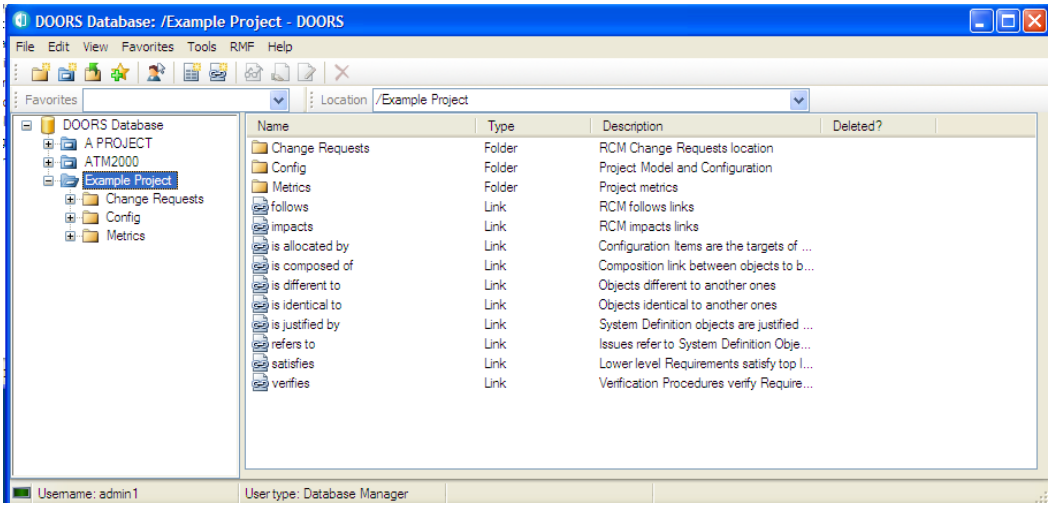


Figure 4 : RMF project creation example

A RMF project contains always a "Config" folder created at the project top level. This folder contains the description of the model and also some RMF configuration information. The other items, link modules, folders, formal modules are depending on each RMF model.

A user must never modify the data into the "Config" folder, and the project administrator should remove all access rights except read (mandatory) for the standard users.

3.1.2 MIGRATE AN EXISTING DOORS PROJECT INTO RMF FORMAT

To transform an existing DOORS project into a RMF project, you have to follow 3 steps:

- Tag the project: Call the menu "RMF ->Create/Tag a RMF project" from the project manager window. Set the option "Tagging an existing project" and browse

the project. Then the options are the same as those described above for project creation.
At this step, your project contains now additional data like a "Config" folder and missing predefined model items.

- Treat existing links: To treat links, you have to find a match between your own *Link Modules* and the *Link Modules* defined by RMF data model (*is justified by*, *refers to*, *satisfies*, *is allocated by*, *verifies,…)*. Then, move Links modules (for expert users only!) by renaming your own *Link Modules* to the RMF ones if match is possible or use the *"Explorer*" RMF tool.
- Treat formal modules: Refer to the paragraph § 3.2.3 MIGRATE A EXISTING DOORS MODULE INTO RMF FORMAT

3.1.3 RMF PROJECT CONFIGURATION

IRDRMFAO allows the definition for the whole project of some parameters that are applied by default in modules, but some of them may also be modified locally for some specific modules.

To configure parameters applicable to the project,

- Open the RMF project,
- Run the menu "RMF ->Configure a RMF Project",

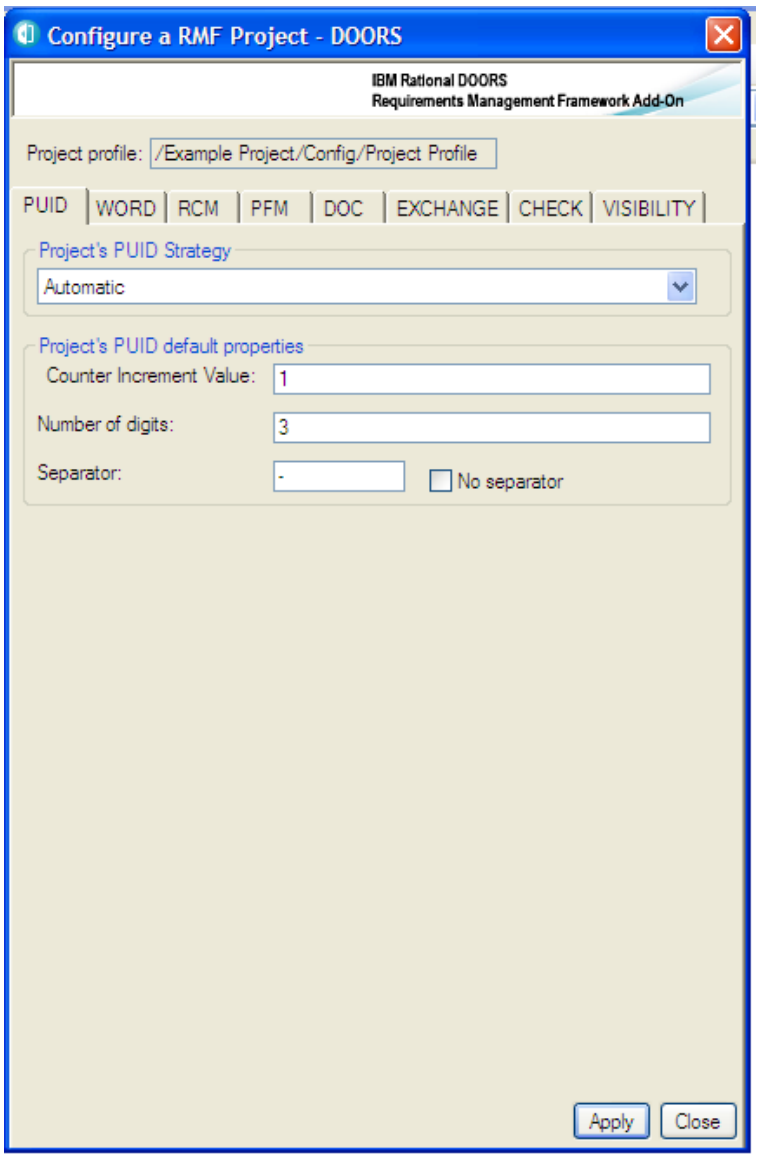


Figure 5 : Project configuration

The different parameters are visible into different tabs.

3.1.3.1 PUID

What is a PUID ?

The PUID means **P**roject **U**nique **Id**entifier. It's the reference name of RMF objects like Requirements.

The two constraints for an identifier are:

- Unicity. Two objects (for example two requirements) should have different PUID values.
- Stability. The PUID should not be modified, even after modify the text or move the object.

You can either decide to manage yourself the PUID entering their values manually or let IRDRMFAO set it automatically. This last case is the default mode called "Automatic PUID strategy". In automatic mode the PUID is composed of 3 parts (Prefix, Object Type and Number) separated by a character. The Object Type is optional in some contexts.

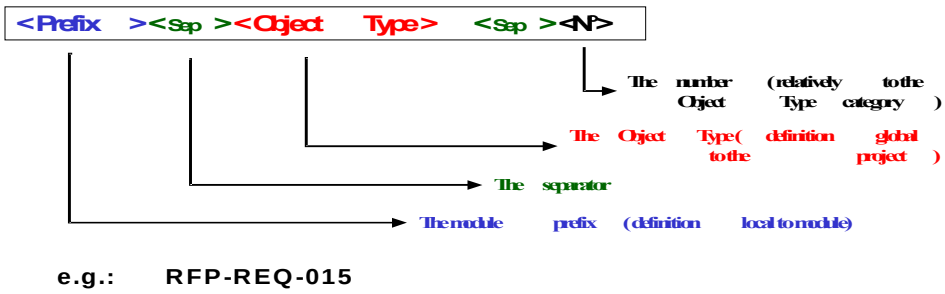


Figure 6 : PUID structure in automatic mode

The manual mode is used only for imported information, when the requirements identifiers are already defined outside of DOORS. It is in general the case only for a few set of modules ("User Requirements" modules for example).

You may also decide to implement the PUID with a DOORS DXL attribute, for example to display the DOORS identifier of an object, if the object is a requirement or some other formal RMF object. In this case the PUID strategy is not taken into account. An example of PUID DXL code is given with IRDRMFAO, in file:

$IRDRMFAO/irdrmfao/misc/attributes/ie_puid.dxl

To configure parameters applicable to the whole project,

- In the project manager window (also called "Database window"), select the RMF project with the DOORS Explorer (left part of the "Database window"),
- Run the menu "RMF ->Configure a RMF project",

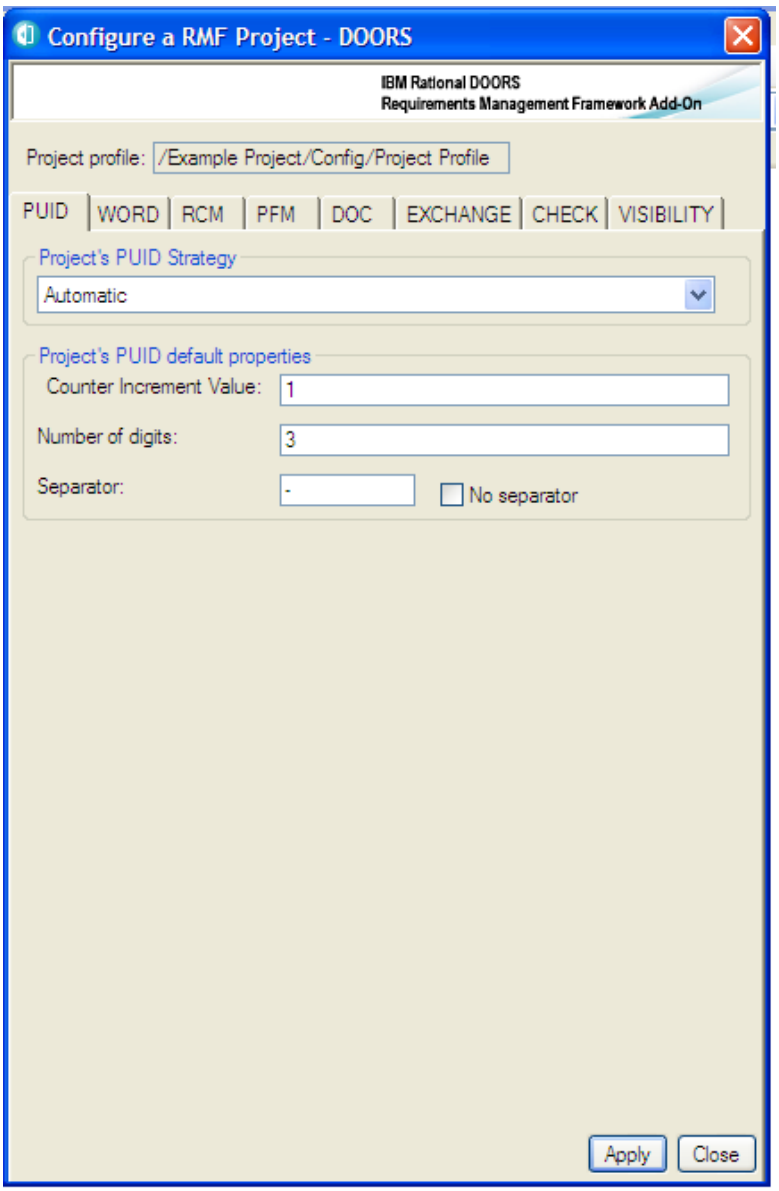


Figure 7 : RMF project configuration, PUID tab

- Select the project PUID strategy : Automatic (default) or Manual,
- Then for Automatic mode, set the PUID properties:
  - Counter increment value (e.g. if 10: RFP-REQ-10, RFP-REQ-20, RFP-REQ-30,…),
  - The number of fixed digit for the counter (e.g. if 3 : RFP-REQ-003, RFP-REQ-020, RFP-REQ-234,…)
    Set "0" if you don't want a number of fixed digit,
  - The separator character. If you want an empty separator you should select the toggle "no separator". If not an empty value will be replaced by a space character.

Remark: In Automatic mode, the Prefix part of the PUID may be defined at module level by the "Module Configuration utility". You can also directly modify it by editing the "Prefix" attribute from the DOORS module properties dialog box.

3.1.3.2 WORD

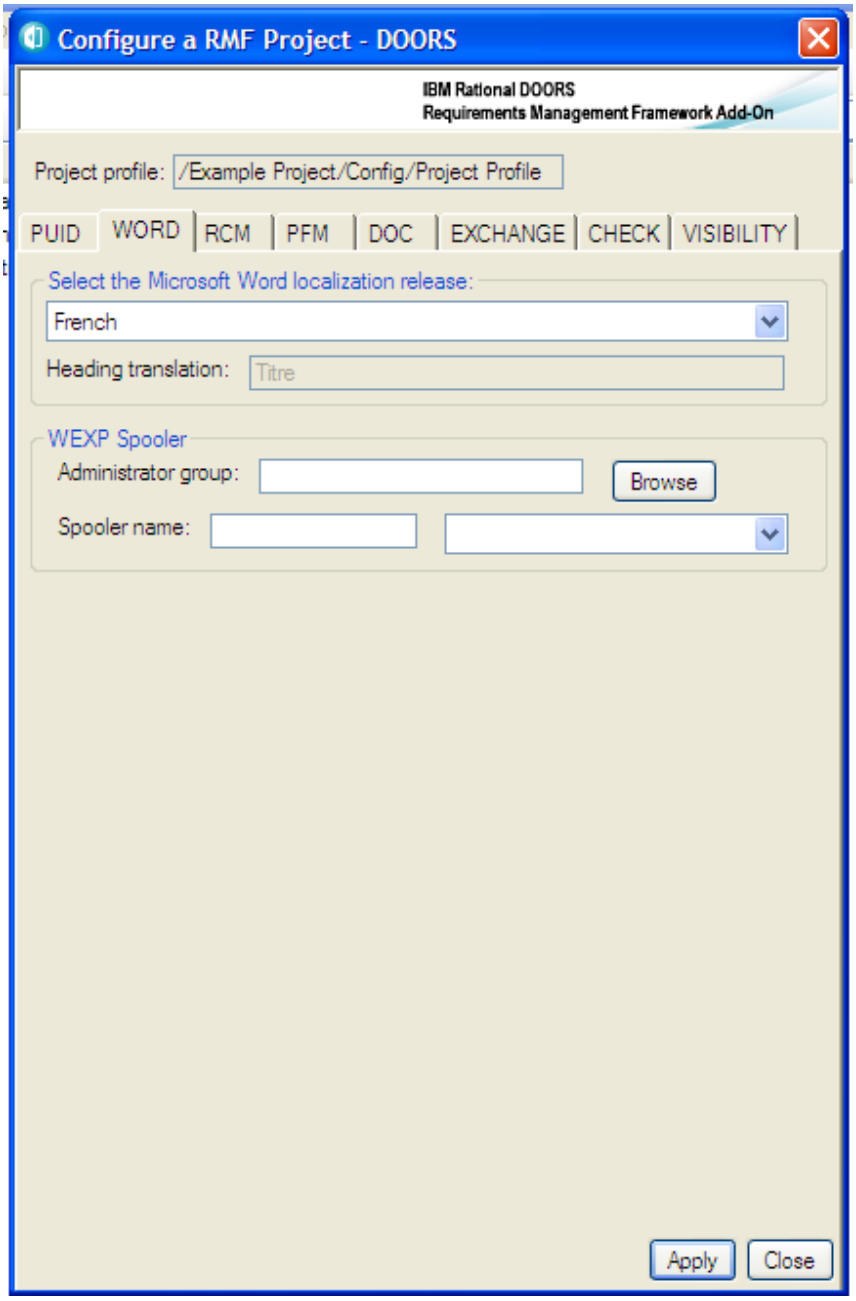


Figure 8 : RMF project configuration, WORD tab

This tab contains some parameters used for the document generation function. The document generation tools of RMF are supporting only Word. You should have Word installed.

- Select the Microsoft Word language. This is used by the document production utility to decide the style name to export for headings (e.g.: "Heading 1" in English, and "Titre 1" in French). Notice that there is no relation between that parameter and the spelling checker to use.

- For the spooler configuration, that can be used to manage WEXP exports on a dedicated computer, you should consult the WEXP manual.

.

3.1.3.3 RCM

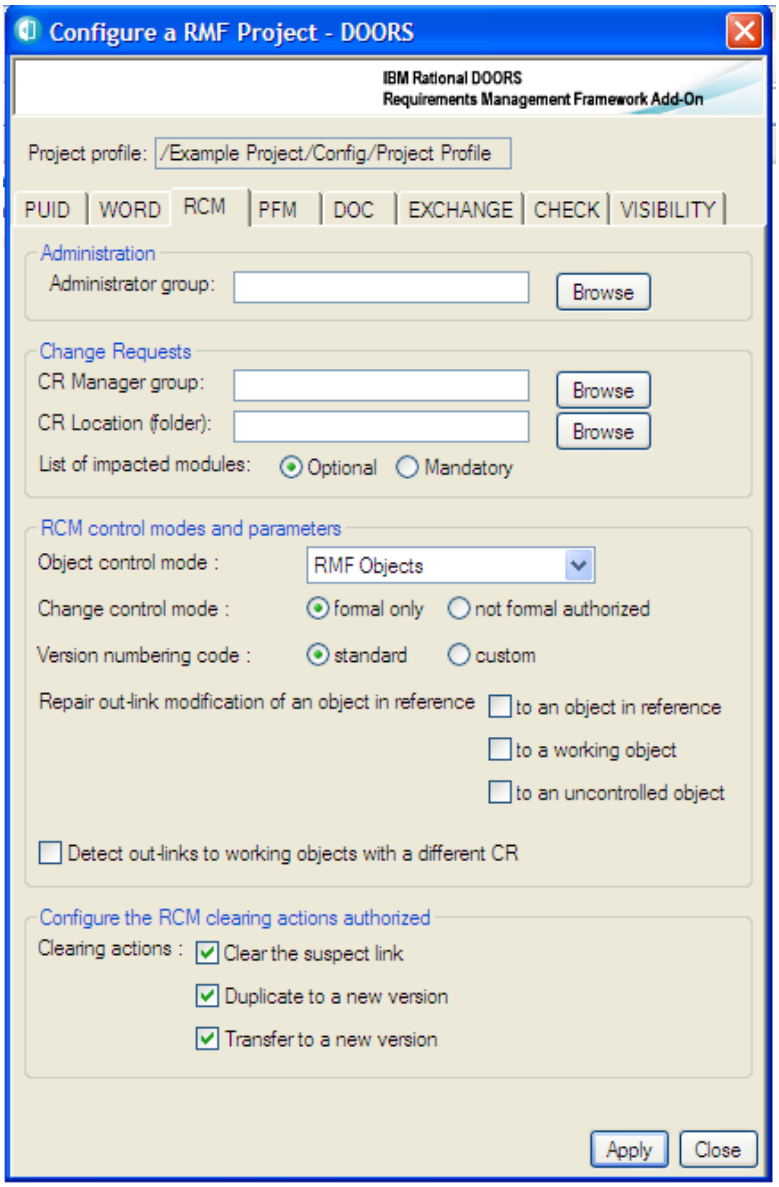


Figure 9 : RMF project configuration, RCM tab

The definition of these parameters is required to deploy the RCM functionality. This configuration is documented into the RCM reference manual.

3.1.3.4 PFM

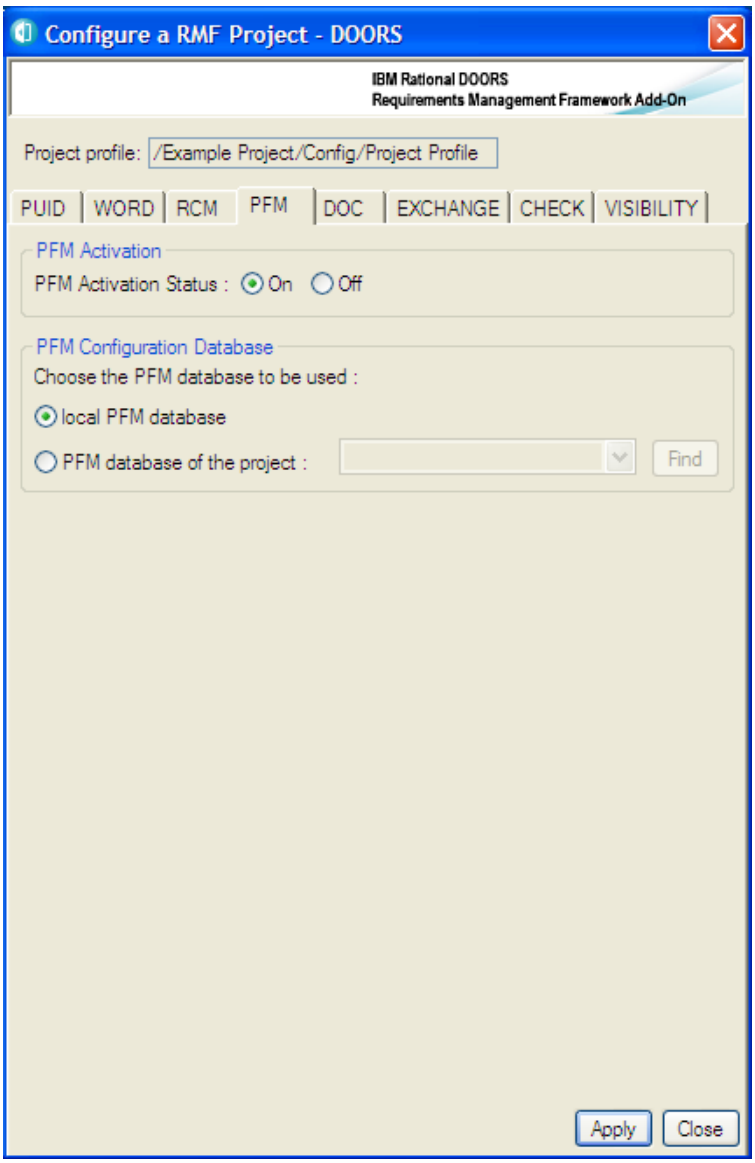


Figure 10 : RMF project configuration, PFM tab

The definition of these parameters is required to deploy the PFM functionality. This configuration is documented into the PFM reference manual.

3.1.3.5 DOC

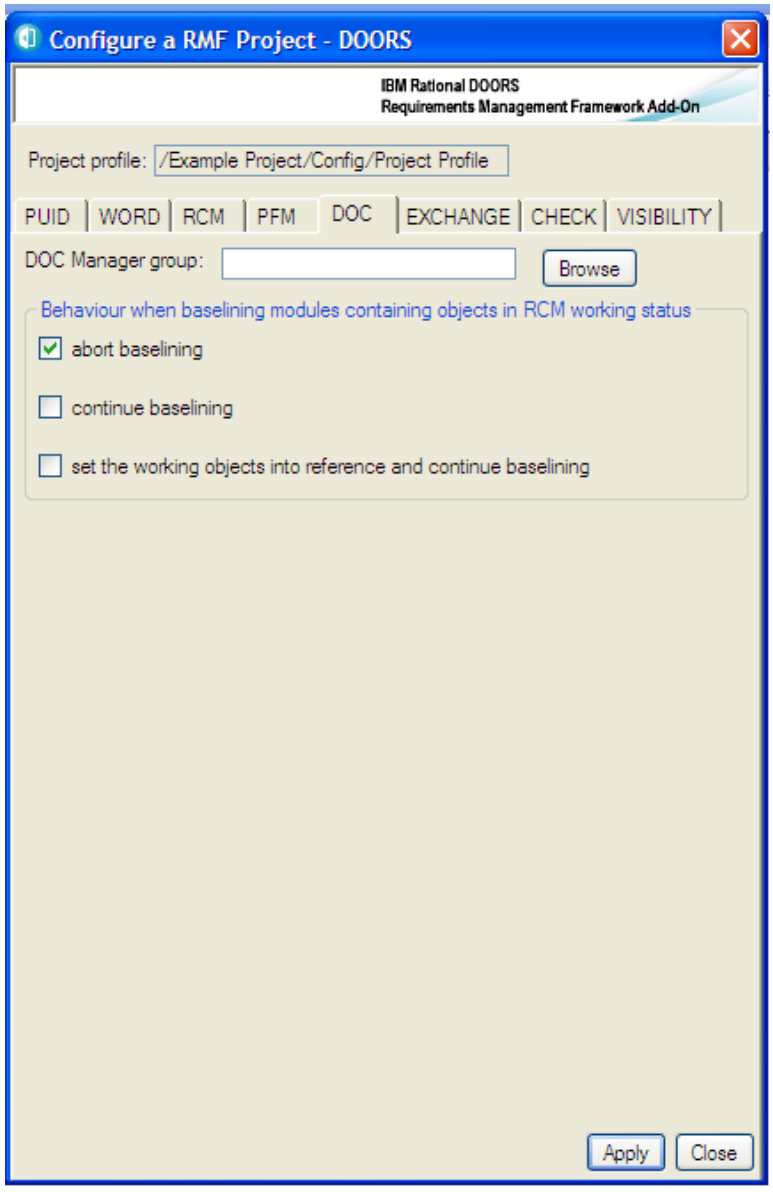


Figure 11 : RMF project configuration, DOC tab

The definition of these parameters is required to deploy the DOC functionality. This configuration is documented into the DOC reference manual.

3.1.3.6 EXCHANGE

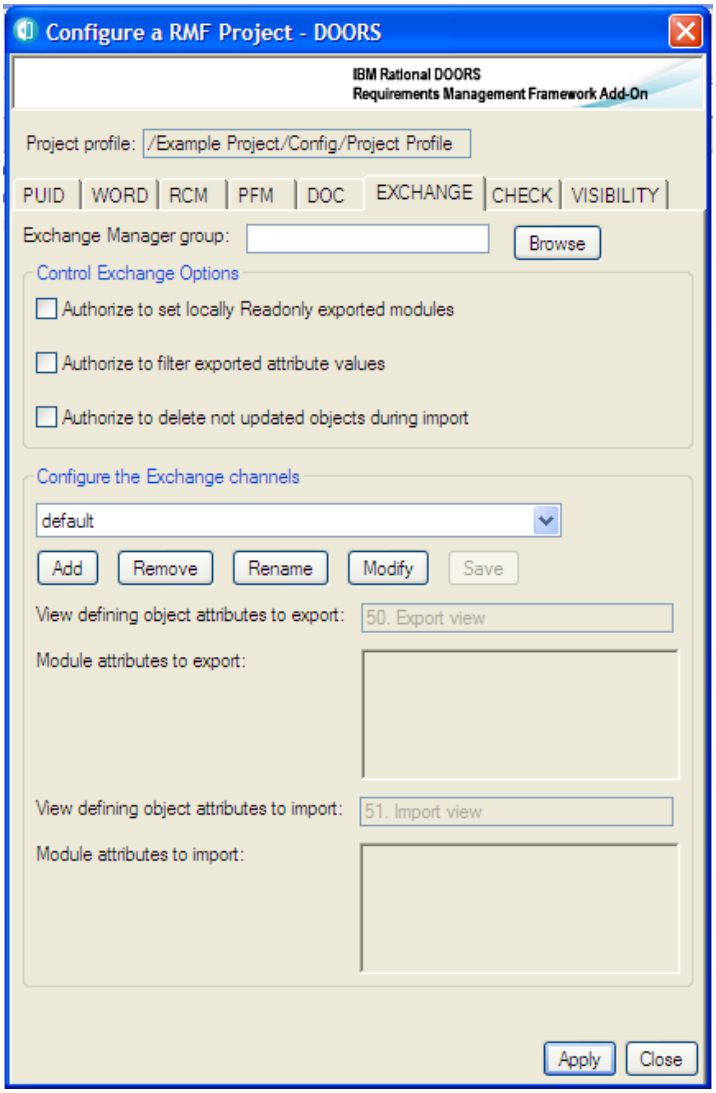


Figure 12 : RMF project configuration, EXCHANGE tab

The definition of these parameters is required to use the Exchange functionality. This configuration is documented into the Exchange reference manual.

3.1.3.7 CHECK

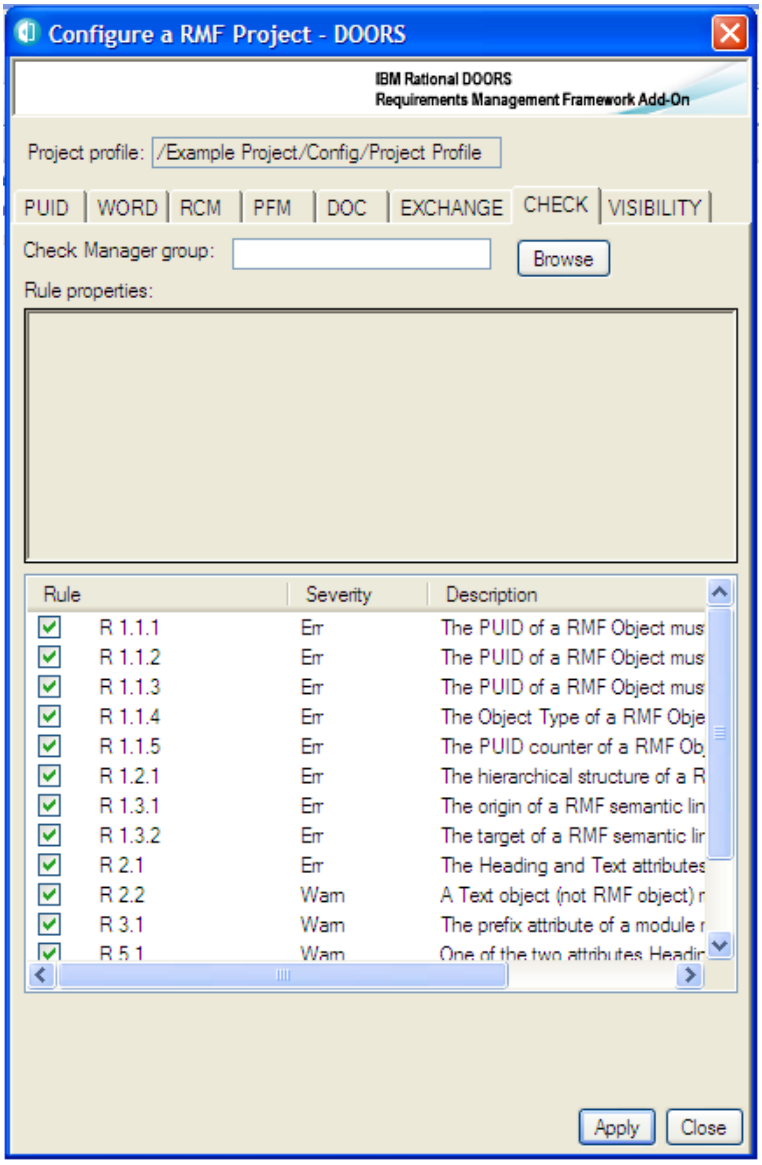


Figure 13 : RMF project configuration, CHECK tab

The definition of these parameters is required to use the Integrity Check functionality.

The Integrity Check functionality is used to verify that the data created into modules are respecting some predefined rules that can be generic or specific.

The first parameter is the definition of the Check Manager group for the project. This role gives the ability to change the Integrity check configuration at module level; it is also used to "protect" the integrity status of each module with specific access rights.

To define the Check Manager click the "Browse" button, and then select a group from the displayed dialog:

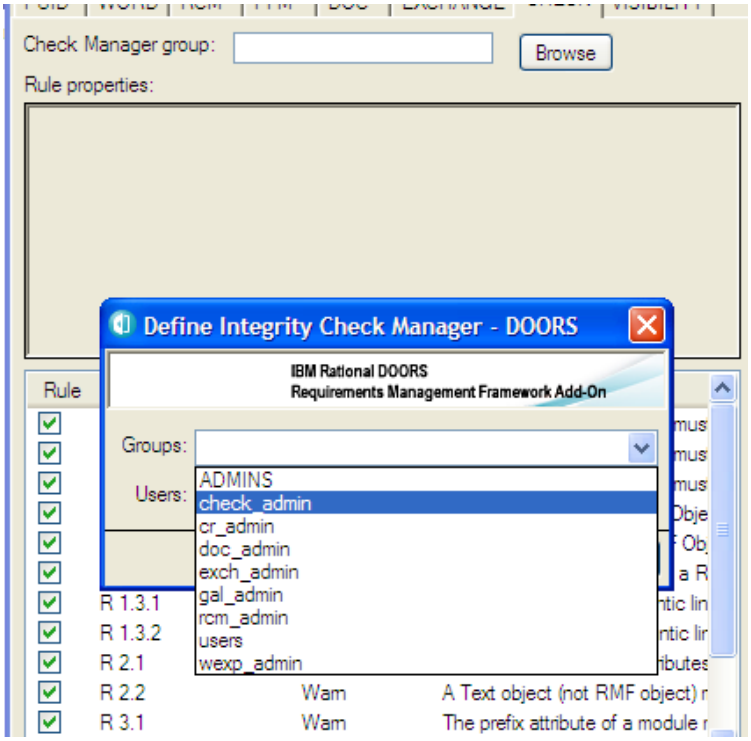


Figure 14 : Defining the check manager

The second parameter is the list of integrity rules that are available with the IRDRMFAO version installed. The integrity rules should be defined with the DXL script language, in some specific place of the IRDRMFAO software. There is a predefined set of rules delivered with IRDRMFAO, these rules are generic and applicable to any RMF project. It is also possible to define rules specific to your model or process, by developing new rules.

When defined, a rule may be activated or not into the project. It is also possible to redefine for some specific modules the set of activated or non activated rules.

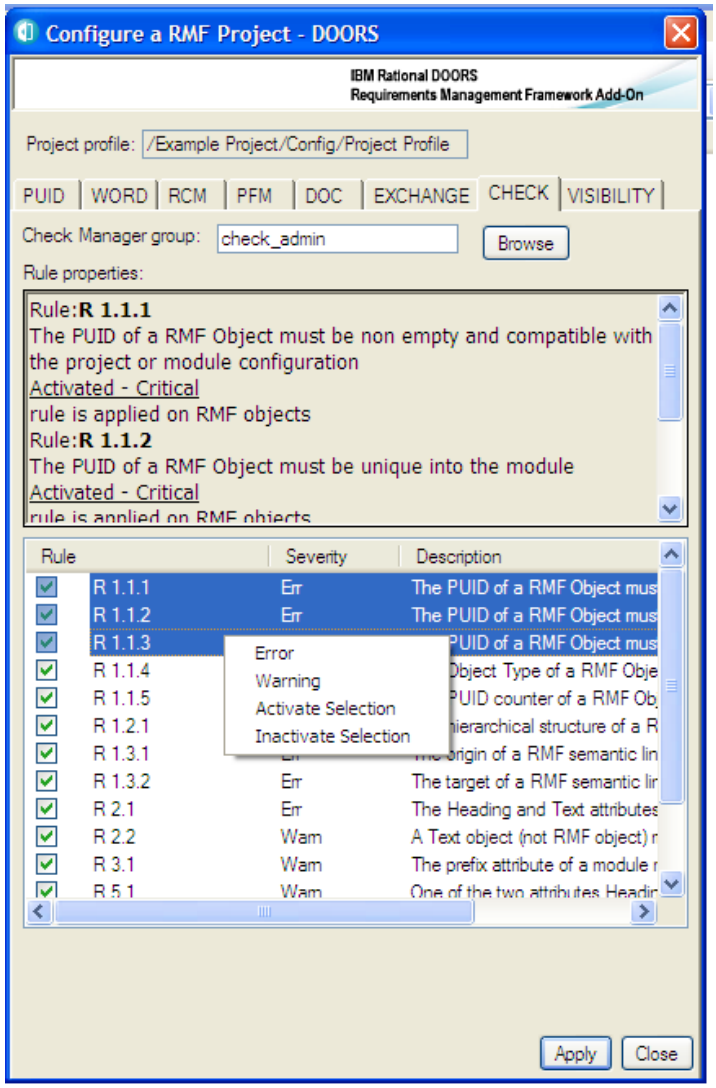


Figure 15 : Activating the check rules

To activate or inactivate a rule, you may check or uncheck the check box associated with each rule, or you may select several rules and apply the operations "Activate selection" and "Inactivate selection" from the contextual menu (right button of the mouse).

The operations "Error" and "Warning" may be used to change the severity level associated with each rule. The initial severity level is defined into the DXL code defining the rule.

When a rule is selected, a description of the rule is displayed in the text field above the rule list.

To get more information on the Integrity Check functionality, refers to chapter § 4.5 CHECK DATA CONSISTENCY.

3.1.3.8 VISIBILITY

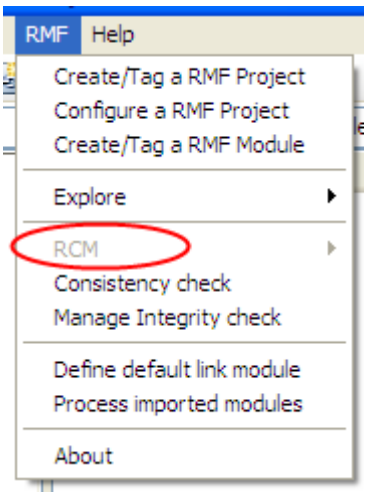
Figure 16 : RMF project configuration, VISIBILITY tab

The visibility is a new concept of RMF 6.0. This configuration should be used to hide for users the non useful functionalities.

For example, the RCM functionality is visible in different places:

- Project menu
- Module menu
- Dialog boxes

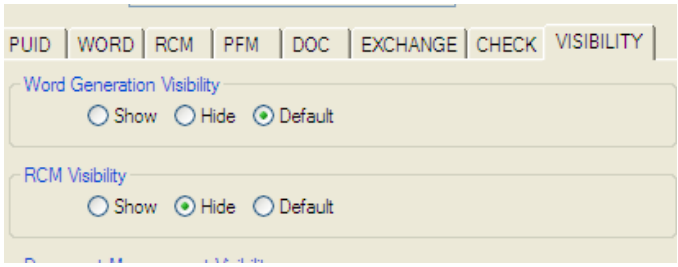
Even if the functionality is not configured, you can see the menus:



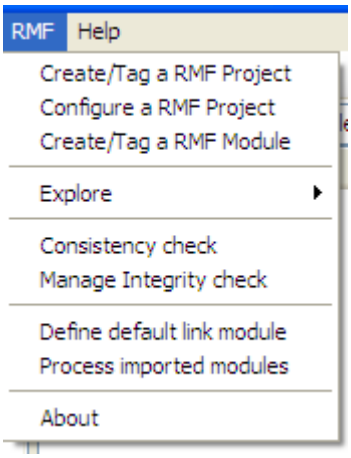
If you hide the functionality in the VISIBILITY tab, all the GUI elements associated with the RCM functionality will be hidden.

Example:

RCM is hidden in the visibility tab



The RCM menu is no more visible:



The RCM configuration tab options are no more accessible.

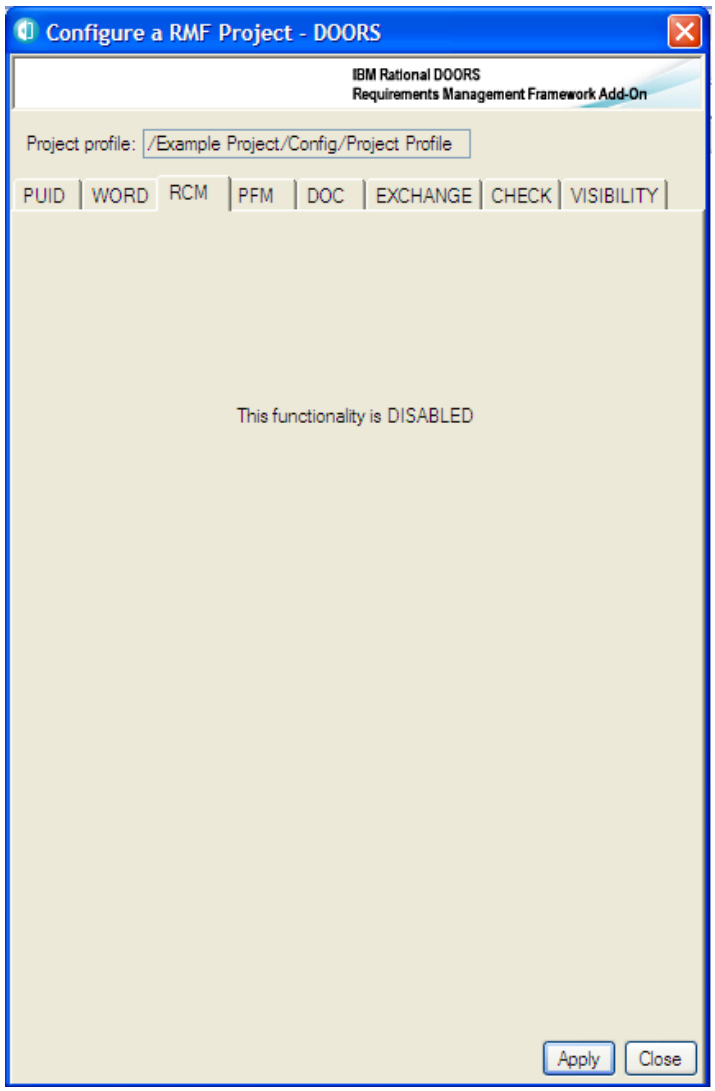


Figure 17 : RMF project configuration, hidden function

The "default" state in the visibility tab is interpreted according to the environment of RMF. RMF checks if an external functionality similar to the RMF functionality is already installed.

For example, if RPE is installed in the DOORS client, there is probably no need to display the Word functionality, because the users are using RPE and not WEXP. But this is only a default behaviour. If RPE is installed, but not used, or if the users want to use the two functionalities, then the RMF administrator may set the option to the "Show" value.

The default behaviour for WORD, RCM, PFM and DASHBOARD can be found into the file $IRDRMFAO/startup/usercallout/fctusercallouts.inc.

The integration that may be tested are:

- WORD: should test the install of RPE
- RCM: should test the install of DOORS/Change integration (TBD)
- PFM: should test the install of DOORS/GEARS integration (TBD)
- DASHBOARD: should test the install of DOORS/INSIGHT integration (TBD)

3.1.4 PROJECT INDEX INITIALIZATION

After having defined the configuration of the project, the project administrator should initialize the Project Index module.

This module is located into the "Config" folder, and is used by some of the RMF tools to accelerate the access to some information. It can be deleted and reinitialized by the administrator at any time.

This operation must be done only after the modification of the access rights of the "Config" folder, not before, because some specific access rights depending on the "Config" should be propagated to this module.

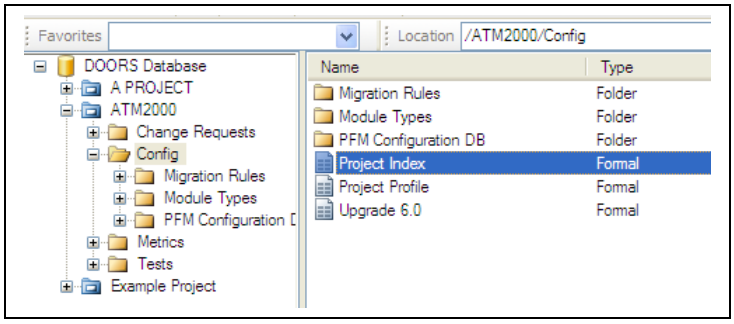


Figure 18 : RMF project index

To initialize it, execute "RMF -> Explore -> Manage Index", and click on the "Full Update" button.

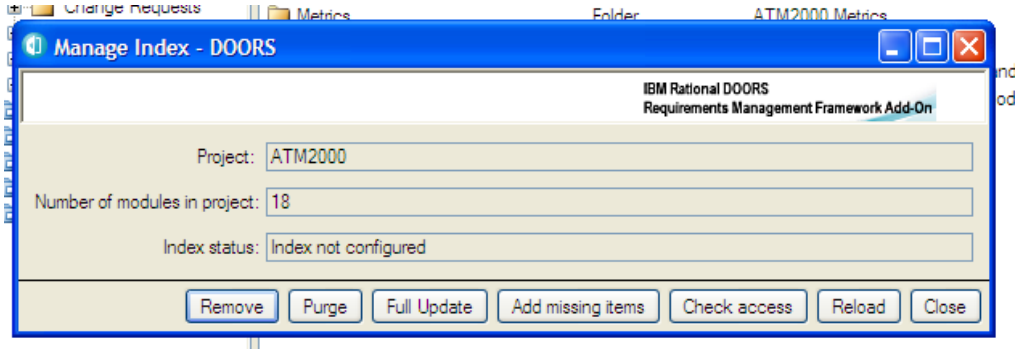


Figure 19 : RMF project index management

After configuration, the information displayed is different :

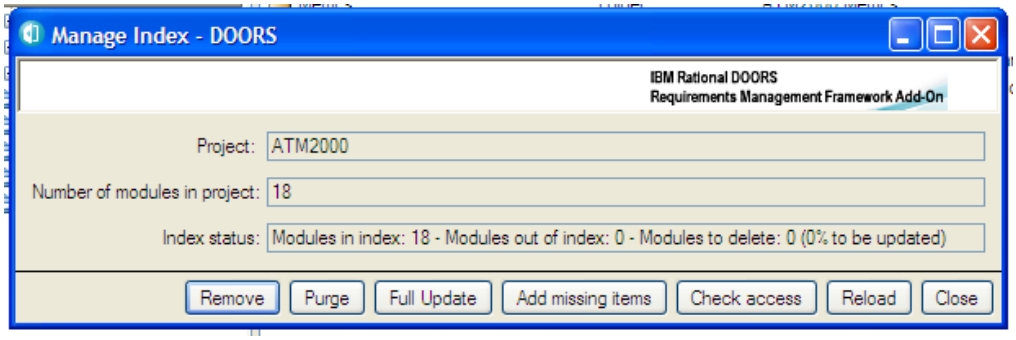


Figure 20 : RMF project index initialized

To get more information on the Project Index module and on the Index Management tool, you should look into the Explorer manual.

3.2 RMF MODULE INITIALIZATION

Your project initialization needs careful thought:

- First, does the generic data model proposed in IRDRMFAO and composed of RMF Module Types (and templates) fit your problem? Which part of the data model may you need? Without having to have a complete answer at the start, you can make a list of the retained modules.
- Second, for each module retained from the data model, are the default attributes proposed relevant for your project? Then, according to your need, you can delete or add attributes within the modules (or modify the model)

If the RMF generic data model or your company data model needs customization see chapter 7 for explanation to what can be done on your project and how.

3.2.1 WHAT IS A MODULE TYPE AND A TEMPLATE ?

The implementation of a RMF data model can be compared as a library of templates.

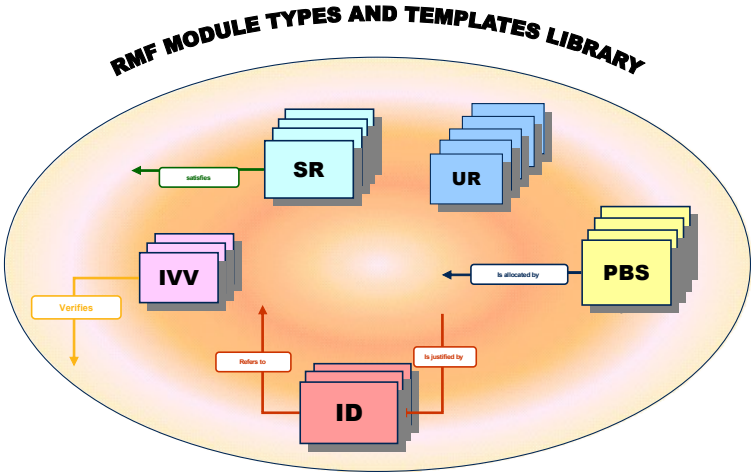


Figure 21 : RMF module types and template library

IRDRMFAO provides different kinds of predefined templates, and each template can be used for several purposes, for example:

- The "IVV" template implements the definitions required to process tests description. It can be used with different roles: "*verification procedure*" module type, "*validation procedure*" module type, "*integration procedure*" module type…and it's not an exhaustive list.
- A module type itself (for example "*verification procedure*") can be instantiated several times in the same project to create different formal modules: test specification of the sub-system A, test specification of the sub-system B …

Example: The "SR" template implements by default the module types "System Requirements", "Subsystem requirements" and "Prime Item Requirements".

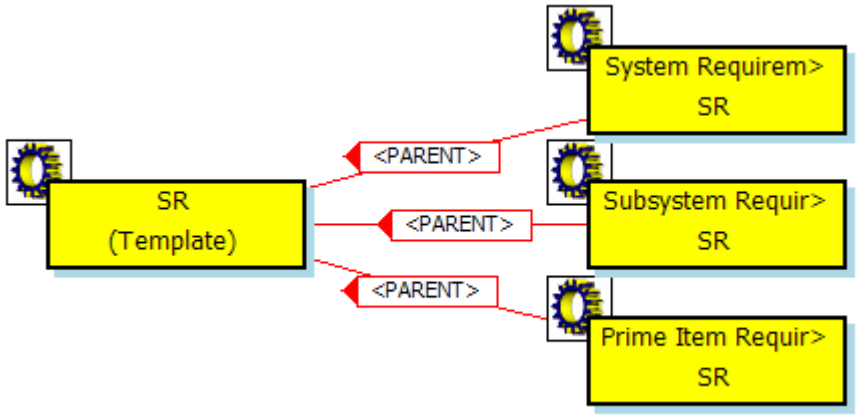


Figure 22 : the SR template

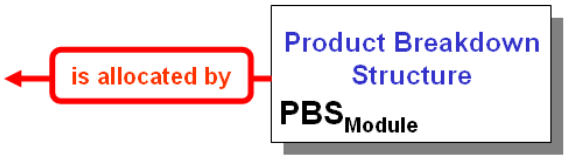
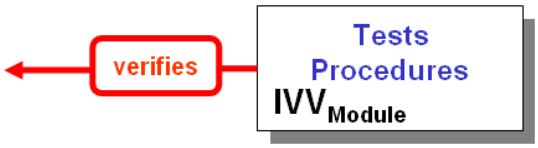
The template defines the nature of the information that is stored into the formal module created from the template, the module types are the different generic usages of the template. Templates and modules types are described into the implementation of a RMF model.

The generic data model, and your own version that is derived from it, can be seen as an assembly of building blocks.

Each building block is implemented by a template supporting different module types, and characterized by its attributes, views and incoming/outgoing links. These are summarized in the table below.

Table 1: List of RMF generic module types

TEMPLATE	MODULE TYPE	USE
UR	User Requirements UR_{Module}	~ **User Requirements Module** *Ex: Contract* *Request For Proposal*
SR	satisfies — System Requirements SR_{Module}	~ **System Requirements Module** *Ex: SSS* ~ **Sub-System Requirements Module** ~ **PIDS Module** *Ex: SRS* ~ Other stakeholders Requirements Modules
ID	refers to — Issues and Decisions ID_{Module} — is justified by	~ **Requirements Analysis Module** ~ **Design Analysis Module**

PBS		~ **PBS Module** *Ex: SSDD*
IVV		~ **Integration Procedures Module** ~ **Verification Procedures Module** ~ **Validation Procedures Module**

For more detail on the modules, consult the appendix.

When implementing a type into the model, two different concepts are used:

The RMF model contains also the definition of some other templates such as DASHBOARD, CR (Change Request) and DI (Document Index). These definitions are not dependent on the data model used by the project and are required by some associated tools (For example RCM is always using the CR template). If you use the functionality, the template must be defined into the model and the predefined definitions of the template must not be modified or removed. You can only add new definitions.

To create your own project, you have to pick from the module type library in order to create your project data, drawing one's inspiration from the generic RMF data model.

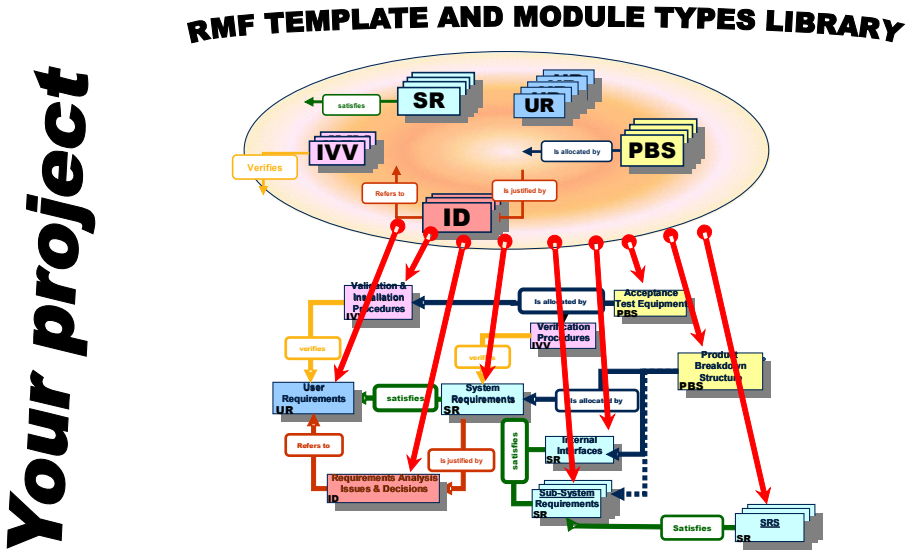


Figure 23 : Build your project from the library

You may examine your RMF Project Model with the **Explorer** tool, by executing the operation "Explore -> Model" from the RMF project menu.

This tool is a project explorer dedicated to RMF. The data processed with the RMF Explorer are displayed and managed according the current RMF model of the project.

Example: partial view of the generic model with the **Explorer** tool

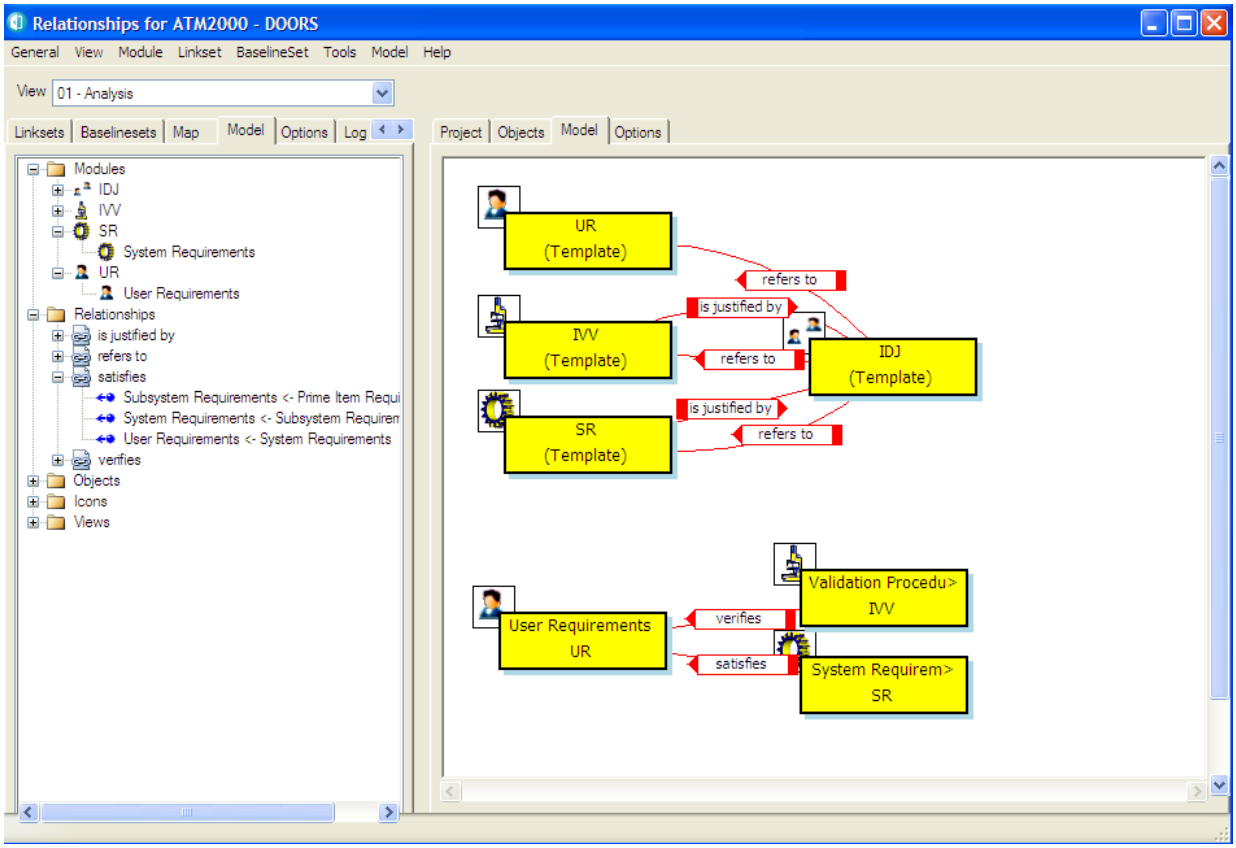


Figure 24 : Explorer (mode with filter)

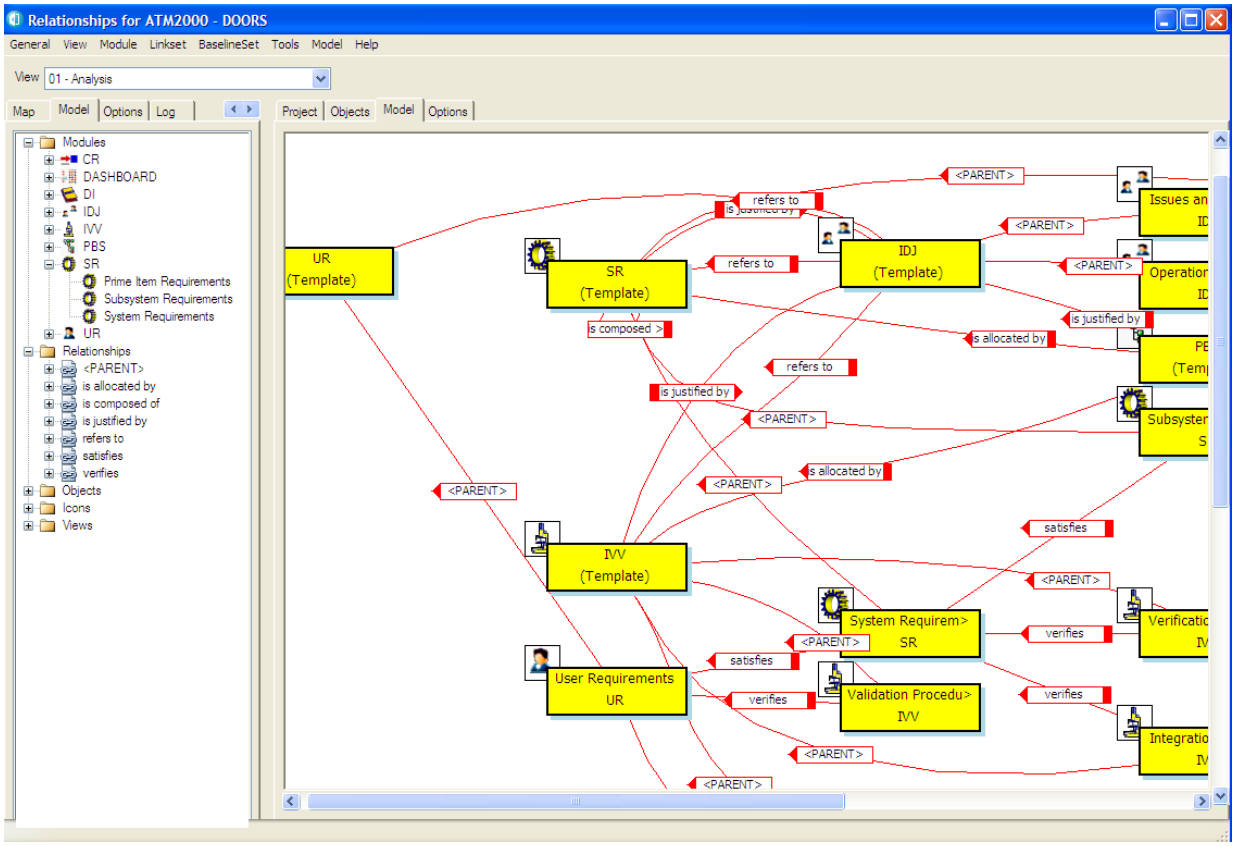


Figure 25 : Explorer (model without filter)

This tool allows you to display different views of your RMF Project Model, according to your process. You may examine for example what is the expected traceability between the different module types and module templates. To have more information about this tool you should read the **Explorer** manual.

The model defines an **abstract data model**, allowing understanding the nature of the information, the project contains the concrete data model.

3.2.2 CREATE A NEW RMF Module

Each time you need to create a module, you have to determine the required module type and template to describe the information contained into the module.

Having done that, you can instantiate the module from the model, by choosing the corresponding RMF type. Then you will have to configure the traceability between this new module and the already existing modules into the project.

Note that for document importation into DOORS, in particular for Microsoft Word document, you can start to export the document (using the specific icon for Word) into DOORS. A DOORS module is then created. In a second step, you have to tag this DOORS module into one of the RMF module types list (see next paragraph).

To create a new RMF module:

- First select its directory location with the DOORS Explorer (left part of the "Database window"),
- Run the menu "RMF ->Create/Tag a RMF module",

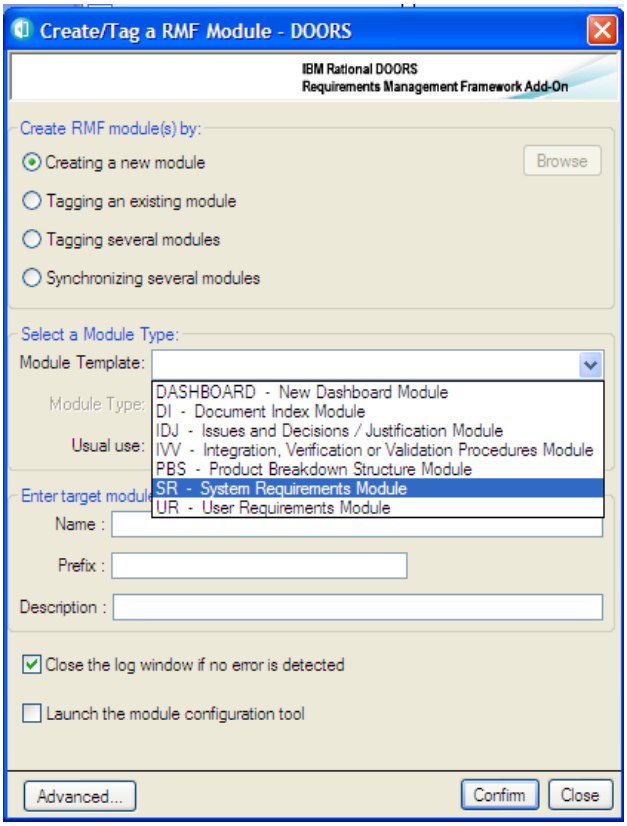


Figure 26: RMF module creation window

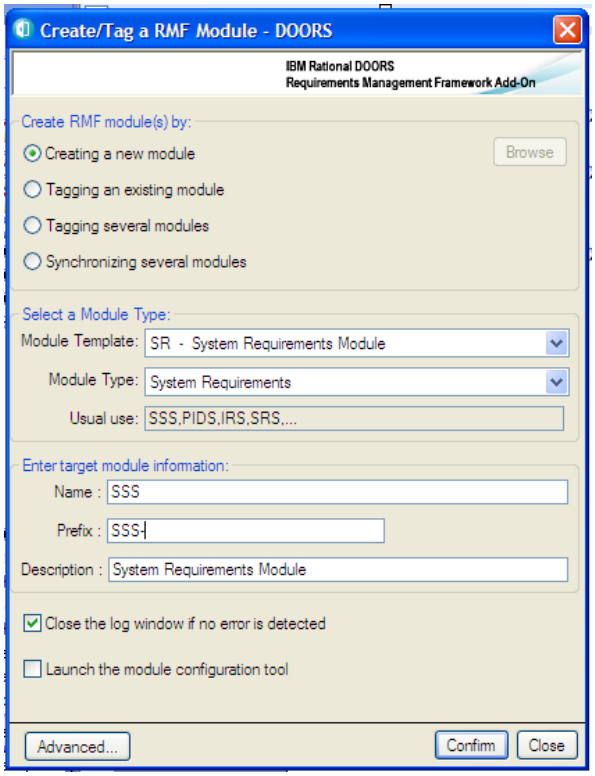
- Select the appropriate template in the list. The followings are example provided by the generic data model:
  - UR for Users Requirements,
  - SR for System Requirements,

- IDJ for Issues & Decisions and Justifications,
- PBS for Product Breakdown Structure,
- IVV for Integration, Verification & Validation,
- DI for Document Index,
- DASHBOARD for dashboard module.

➢ Select the appropriate module type amongst the list of available types according to the selected template

➢ Give a name, a prefix and a description to the new module. Only name field is mandatory, RMF will put a default description if the field is left empty. Note that the module name and description can be modified later.

➢ Click on the "Confirm" button, if successful, an acknowledgement window is then displayed.



The tool supports also other functionalities:

- **Tagging an existing module**: to add the RMF definitions of the selected module type to an existing module. Can be used also to upgrade a module after an evolution of the model.
- **Tagging several modules**: to add the RMF definitions of the selected module types to a set of modules.
- **Synchronizing several modules**: to upgrade a set of modules after an evolution of the model.

The **Advanced** button allows the selection of the definitions to add to the module to tag.

You may also create a new module from the **Explorer** tool, by calling the operation "Create module" from the selected type:

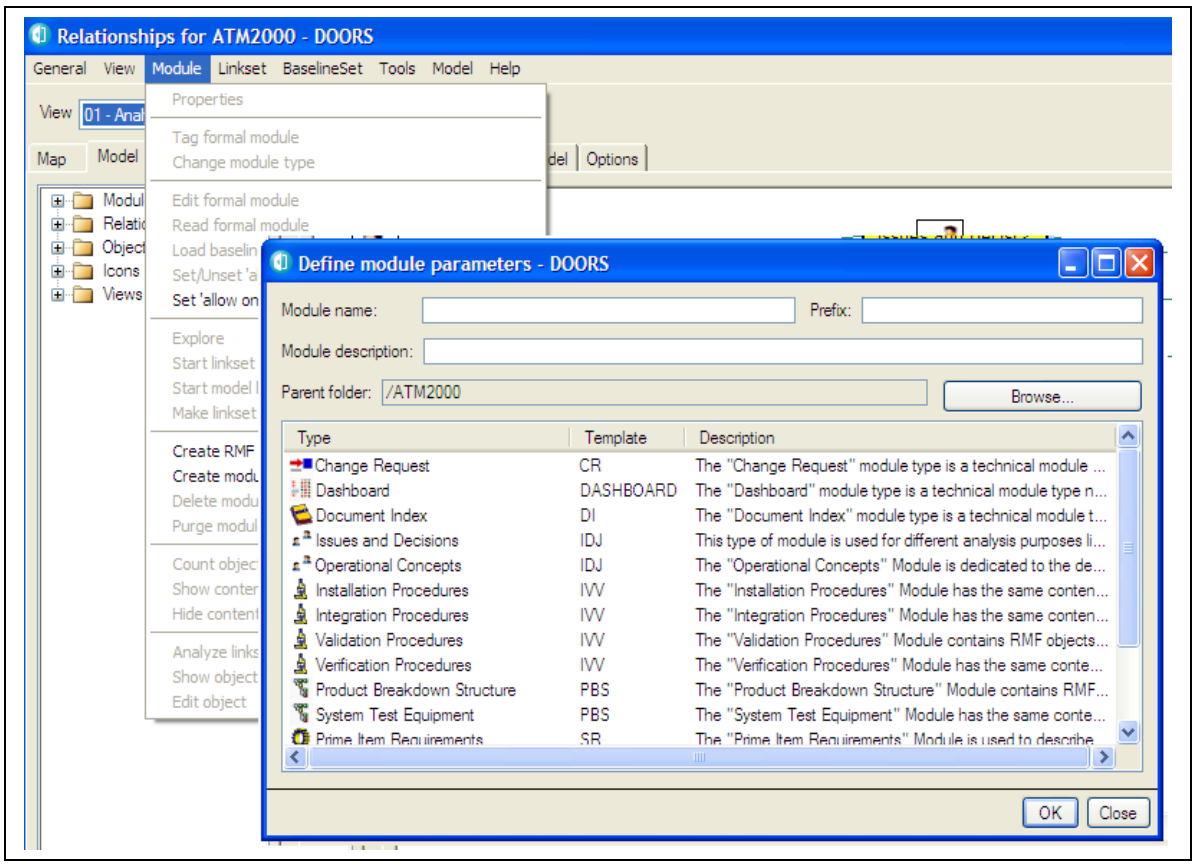


Figure 27 : Create module in Explorer

You may specify the folder in which you want create the module with the "Browse" button. The module type is automatically set to the right value.

At this point, you can create the default linkset pairing making use of a DOORS feature (refer to § 3.3 DEFINE THE DEFAULT LINKSET PAIRING).

3.2.3 MIGRATE A EXISTING DOORS MODULE INTO RMF FORMAT

To migrate an existing DOORS module to a RMF module derived from a module type defined into the model, you have to proceed with the same manner as described in the previous paragraph for RMF module creation, but this time select the toggle "Tagging an existing module" and browse to find the module to tag.

Alternatively you can also:

- Select a module in the DOORS database browser interface
- Launch the "RMF -> Create/tag a RMF module" command

This way, the option "Tagging an existing module" is already selected and you don't need to browse the module to tag.

Example:

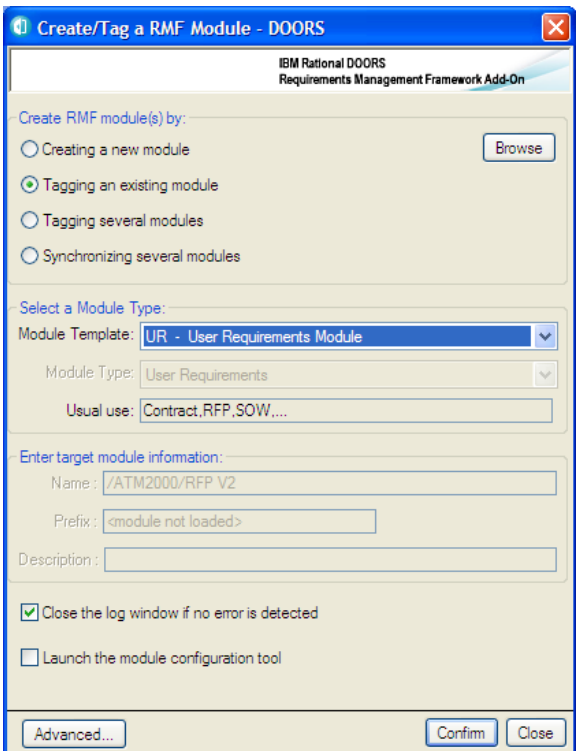


Figure 28 : Create/Tag a RMF Module

An alternative way is the call of the operation "Tag module" in the **Explorer** tool:

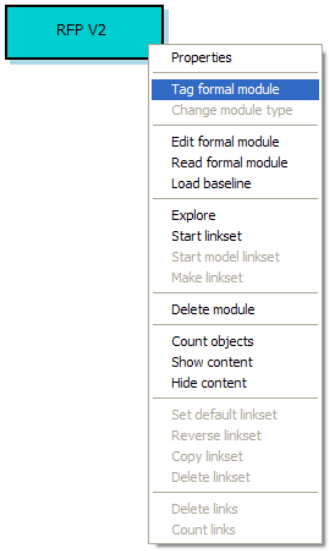


Figure 29 : Tag module in Explorer

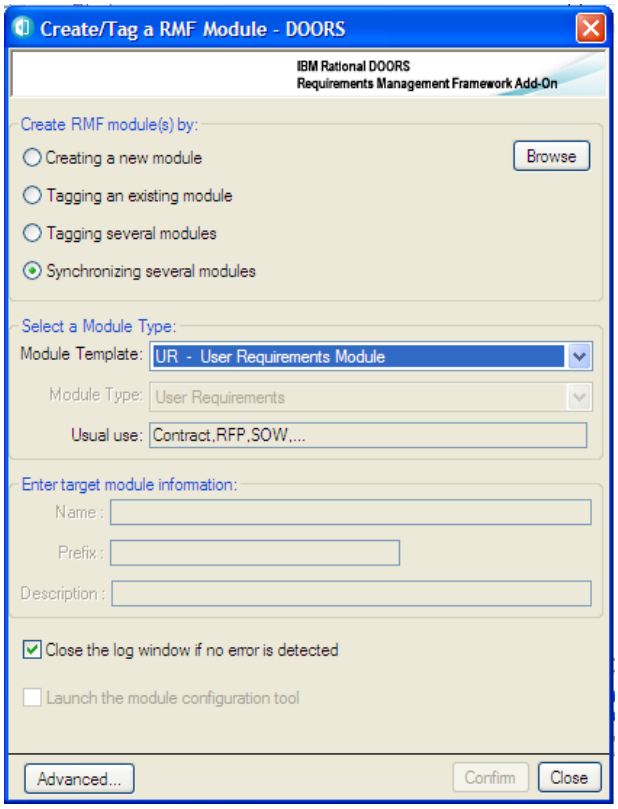
All the definitions associated with the selected module type are added to the target module.

3.2.4 MIGRATE SEVERAL EXISTING DOORS MODULES INTO RMF FORMAT

This is also possible by using the third option "Tagging several modules":

- Launch the "Create/tag a RMF module" command
- Select the "Module Type" value in the list

- Select the "Tagging several modules" option



- Click the "Browse" button, the following dialog box appears:

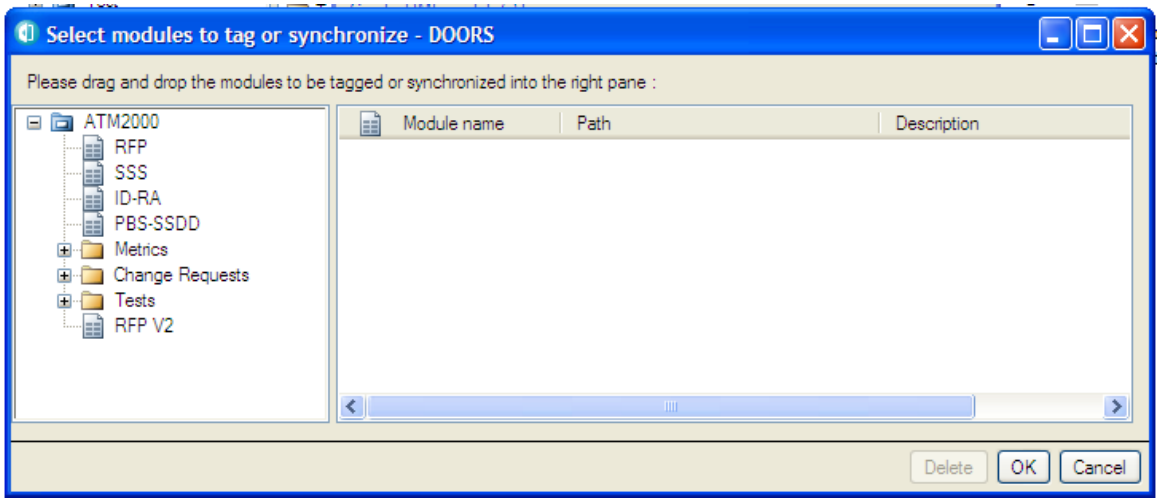


Figure 30 : Browse dialog (1) in Create/Tag a RMF module

Select the modules that you want to synchronize with the same module type by dragging and dropping them in the right pane:

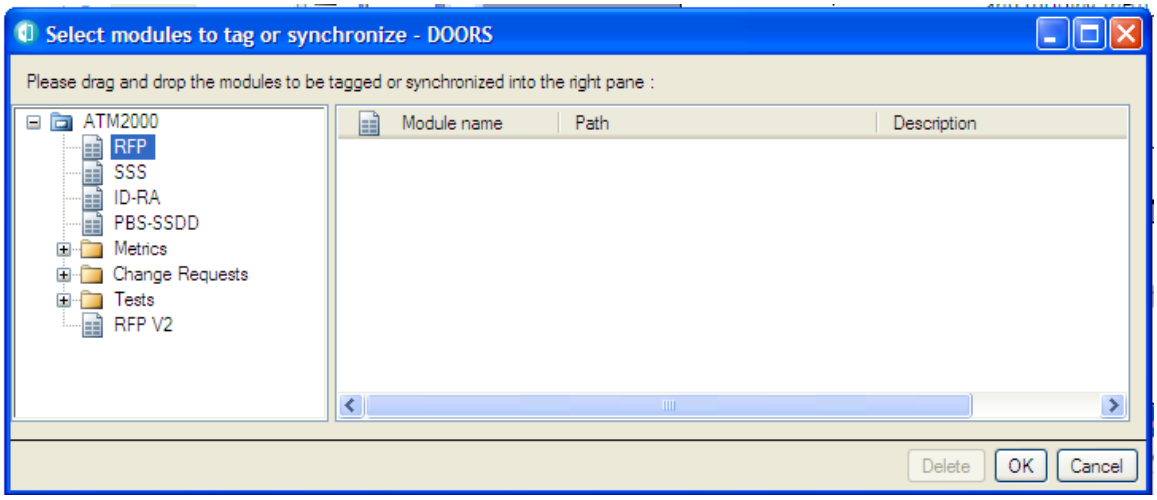


Figure 31 : Browse dialog (2) in Create/Tag a RMF module

The dragged module shows in the right pane, as follows:

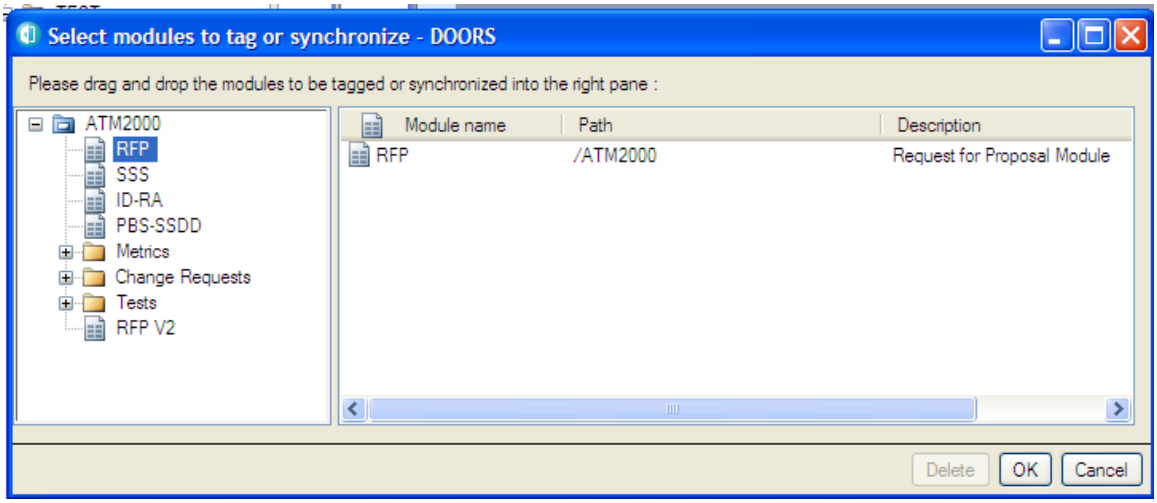


Figure 32 : Browse dialog (3) in Create/Tag a RMF module

Then click on the "OK" button. The browse dialog disappears.

Select the module type to use as a model and then click on the "Create" button of the main dialog box.

You may drag several modules in only one operation by dragging a folder or a project: all the modules contained by the folder or the project are automatically dragged.

The operation "Synchronizing several modules" has a different behaviour: when selecting this mode, the tagging operation is applied only to the modules that have already been tagged with the selected type. This mode should be used to propagate an evolution in the model (new attributes or new views) to the already existing modules.

3.2.5 RMF MODULE CONFIGURATION

IRDRMFAO allows the definition of some parameters that are applied at module level. Some are defined at project level but you may modify them locally.

To configure parameters applicable to a particular RMF module,

- Open the module,
- Run the menu "RMF ->Configure Module",

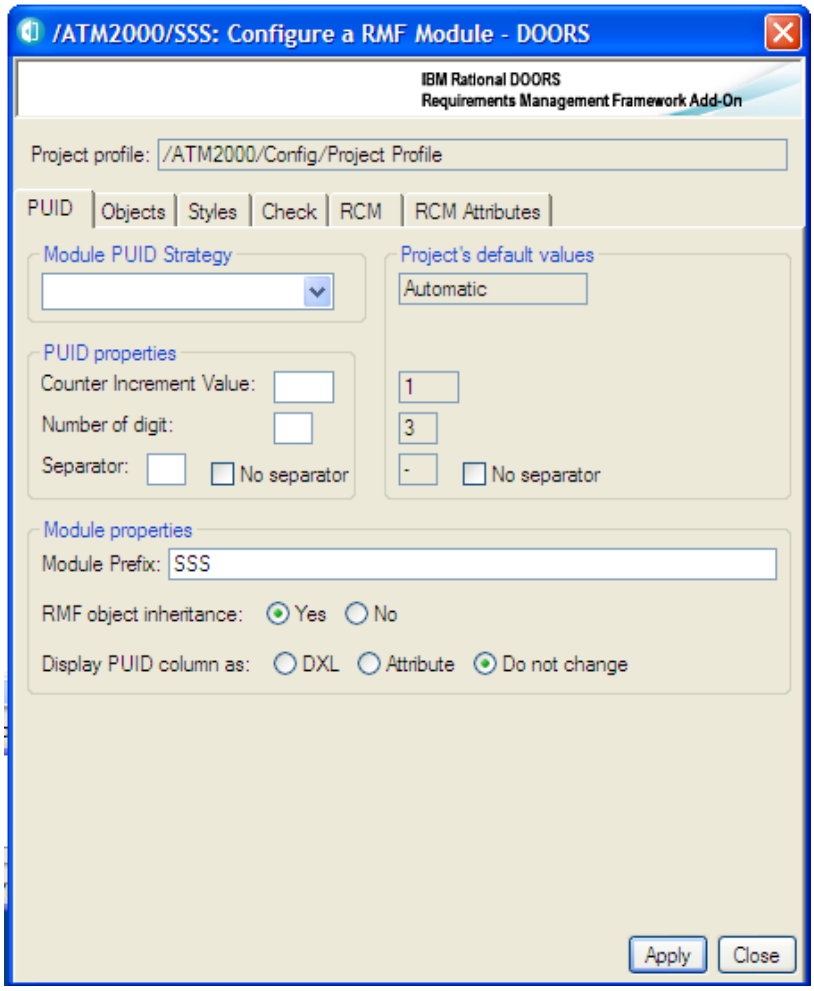


Figure 33 : RMF module configuration window

The different items of the module configuration are visible into different tabs.

3.2.5.1 PUID

This tab contains all the PUID configuration options.

Top part of the window : concerns overridable project parameters. Refers to § 3.1.3 RMF PROJECT CONFIGURATION to know more about PUID meaning, setting strategy and properties.

Module Prefix: In Automatic mode, defined the Prefix part of the PUID (Refers to § 3.1.3 RMF PROJECT CONFIGURATION).

RMF Object inheritance: "Yes" makes the attribute "IE Object Type" to be inherited… by default choose this option except if you want to make some chapter object to be identified (not recommended in general, but useful for PBS or IVV modules to describe hierarchies of RMF objects).

Display PUID column: "DXL" option allows preventing direct modification of PUID displayed in views whereas "Attribute" option allows it. All the views of this module are going to be checked and modified. The option "Do not change" let the vieliws as they are.

3.2.5.2 Objects

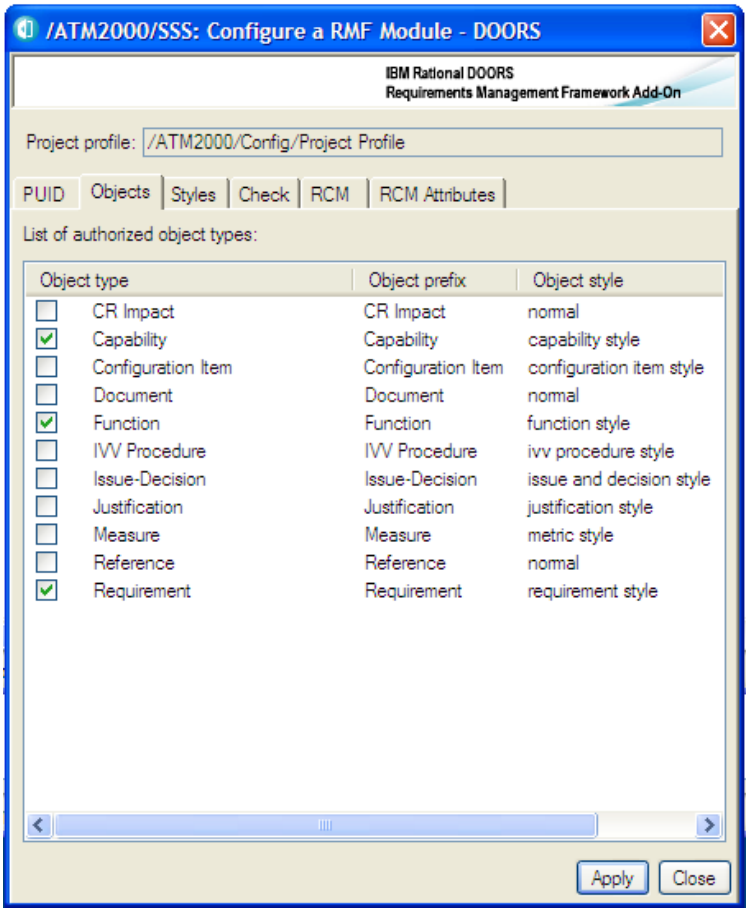


Figure 34 : RMF module configuration, Objects tab

This is the list of the object types that you can identify. This list is created from the list of all object types declared into the RMF model of the project. By default, the checked boxes are those allowed according to the module type. It can be modified by checking/unchecking the boxes. Frequently, only one object type is used into a given module type.

3.2.5.3 Styles

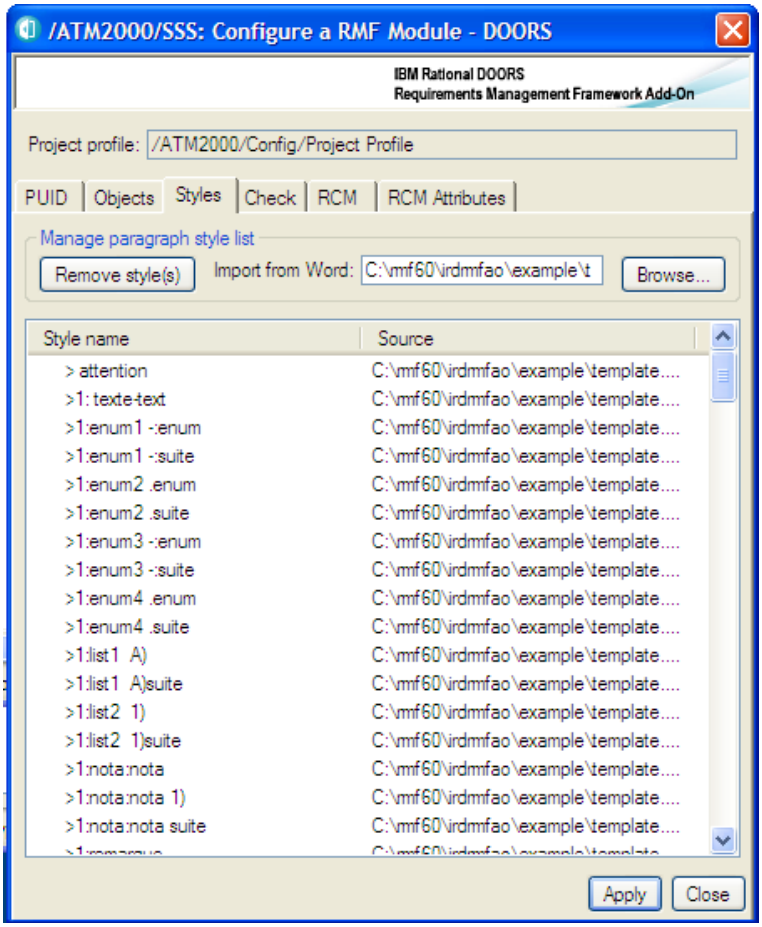
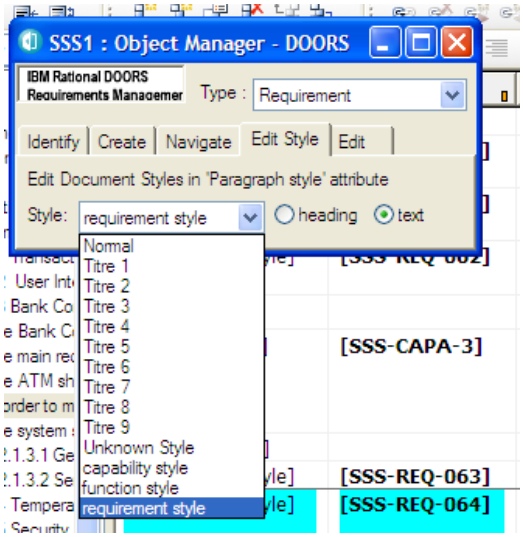
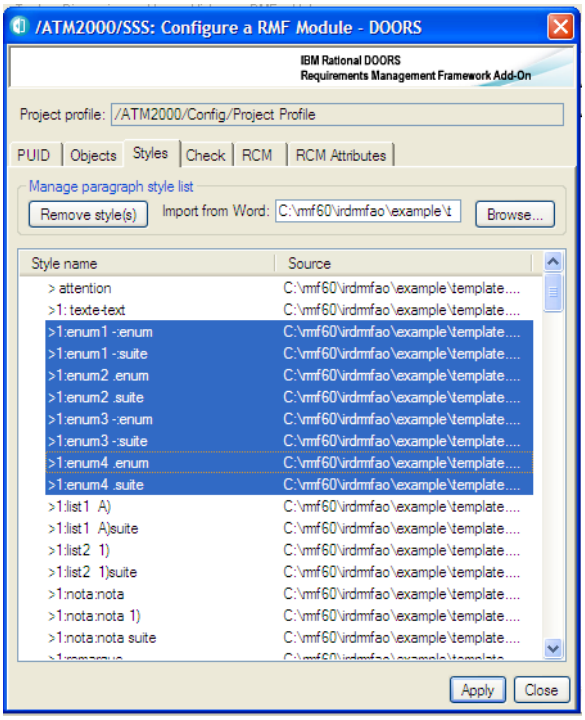


Figure 35 : RMF module configuration, Styles tab

You can use the browse operation to pick all paragraph styles defined into a document template. The styles are then usable into the **Manage Object** dialog to associate a Word paragraph style with an object text.

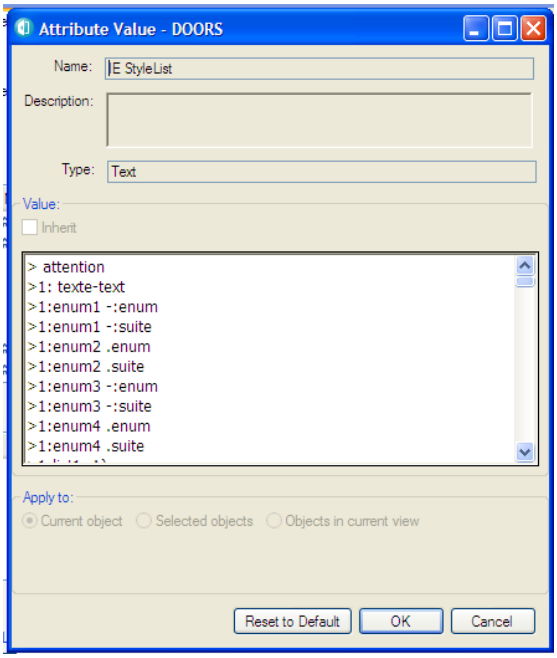


You may also remove some defined styles by selecting the style names into the list, and then click the “Remove style(s)” button.



The list contains only the style name, the style definition is only in the Word template, and the styles must be defined in the templates used by the document generation.

This information is saved into the module attribute “IE Style List”:



3.2.5.4 RCM and RCM attributes

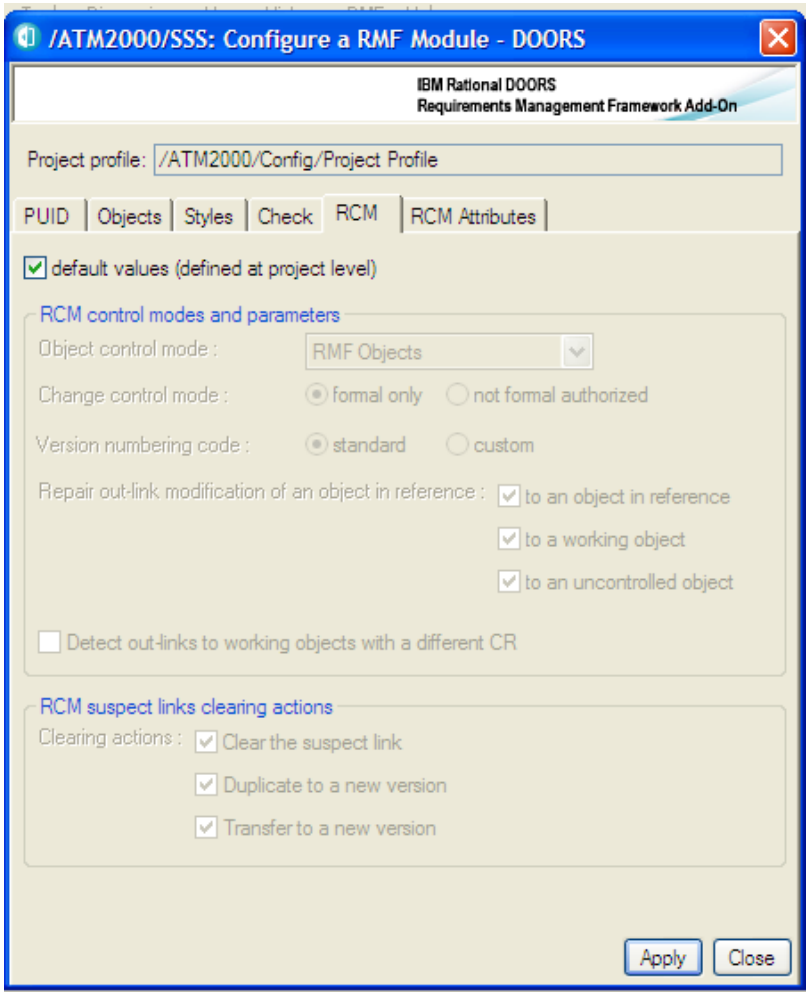


Figure 36 : RMF module configuration, RCM tab

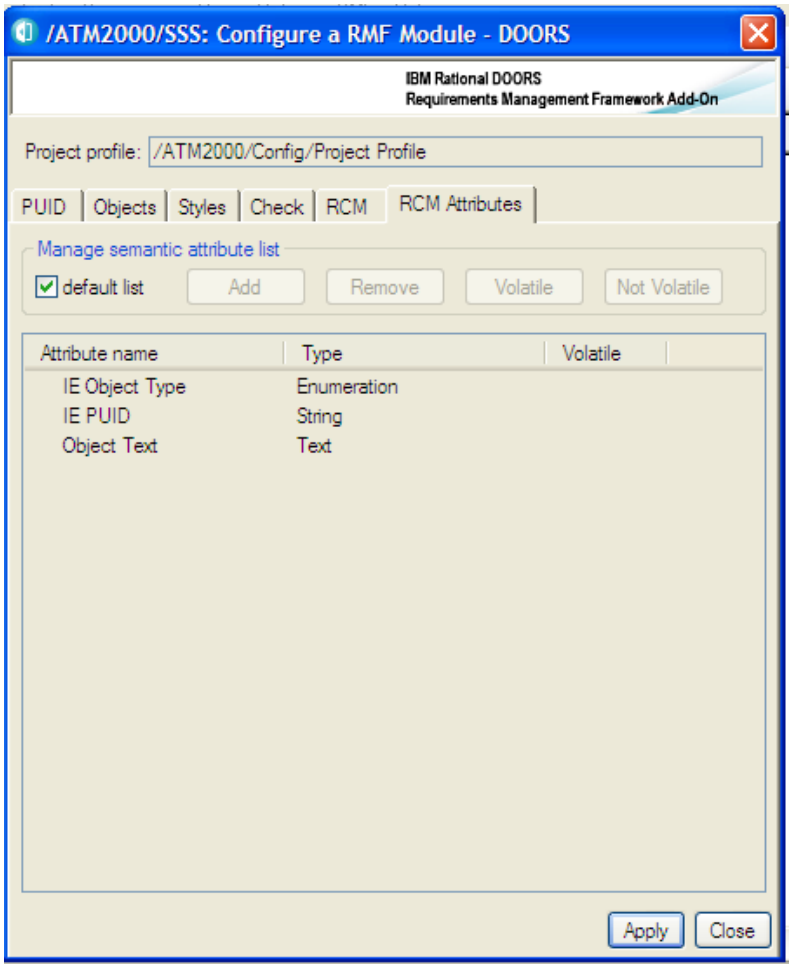


Figure 37 : RMF module configuration, RCM attributes tab

You should consult the RCM documentation to understand the use of these parameters. They are accessible only if RCM is initialized into the project and if the module is under RCM control. Only a user defined into the project as a RCM administrator is able to modify these parameters.

3.2.5.5 Check

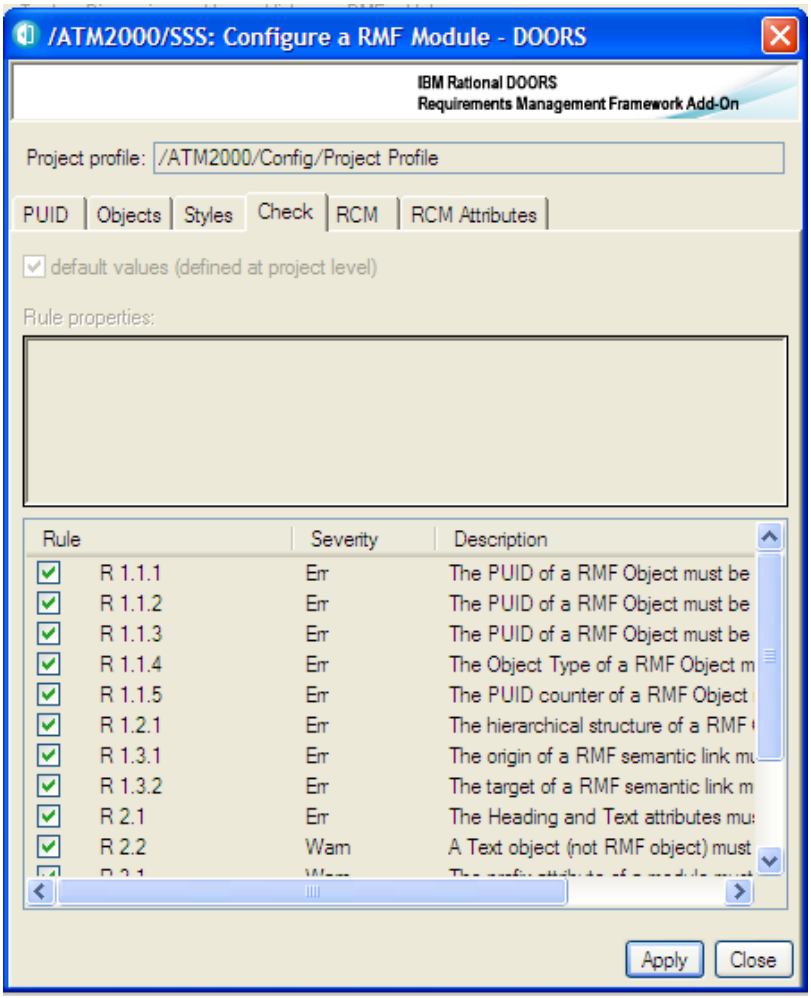


Figure 38 : RMF module configuration Check tab

This window can be used to modify the Integrity Check configuration specifically for a module. It is possible only:

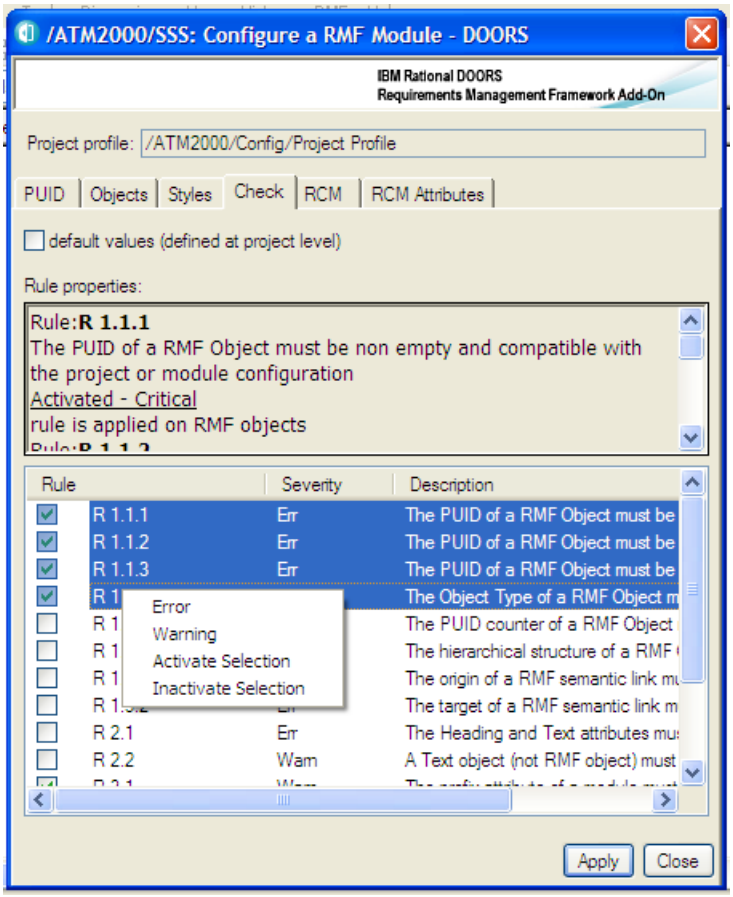
- Integrity Check is already configured at project level
- You have the role Integrity Check Manager for the current project

To define the Integrity Check configuration you must first uncheck the toggle "default values" at the top of the window.

The configuration dialog allows you to:

- Activate or inactivate a rule. By default the activation state is inherited from the project level activation state
- Change the severity level of the rule. The severity level may be Error or Warning. By default it is inherited from the project level severity level.

You may select one or several integrity rules, and use the operations from the contextual menu (right button of the mouse) to change the status of a rule, or you can check or uncheck the check box associated with each rule to activate or inactivate the rule.



To get more information on the Integrity Check functionality, refers to chapter § 4.5 CHECK DATA CONSISTENCY.

3.3 DEFINE THE DEFAULT LINKSET PAIRING

Once, you have created several modules in your project, you have to teach to DOORS which link types should be used by default (or be prohibited !) between pairs of module: DOORS allows this (this is not a IRDRMFAO feature). For each pair of modules, you have to define the default link modules between this module and the others that are used whenever anyone creates a link between them.

To do this with DOORS operations, run "File->Module properties…", select "Linksets" tab and add all the default linkset pairings (only the outgoing links from that current module). Remember to define all links in the "upward" direction.

Remember that when using IRDRMFAO, you should define this configuration in accordance with the RMF model that describes the possible relationships between the different module types.

Example:

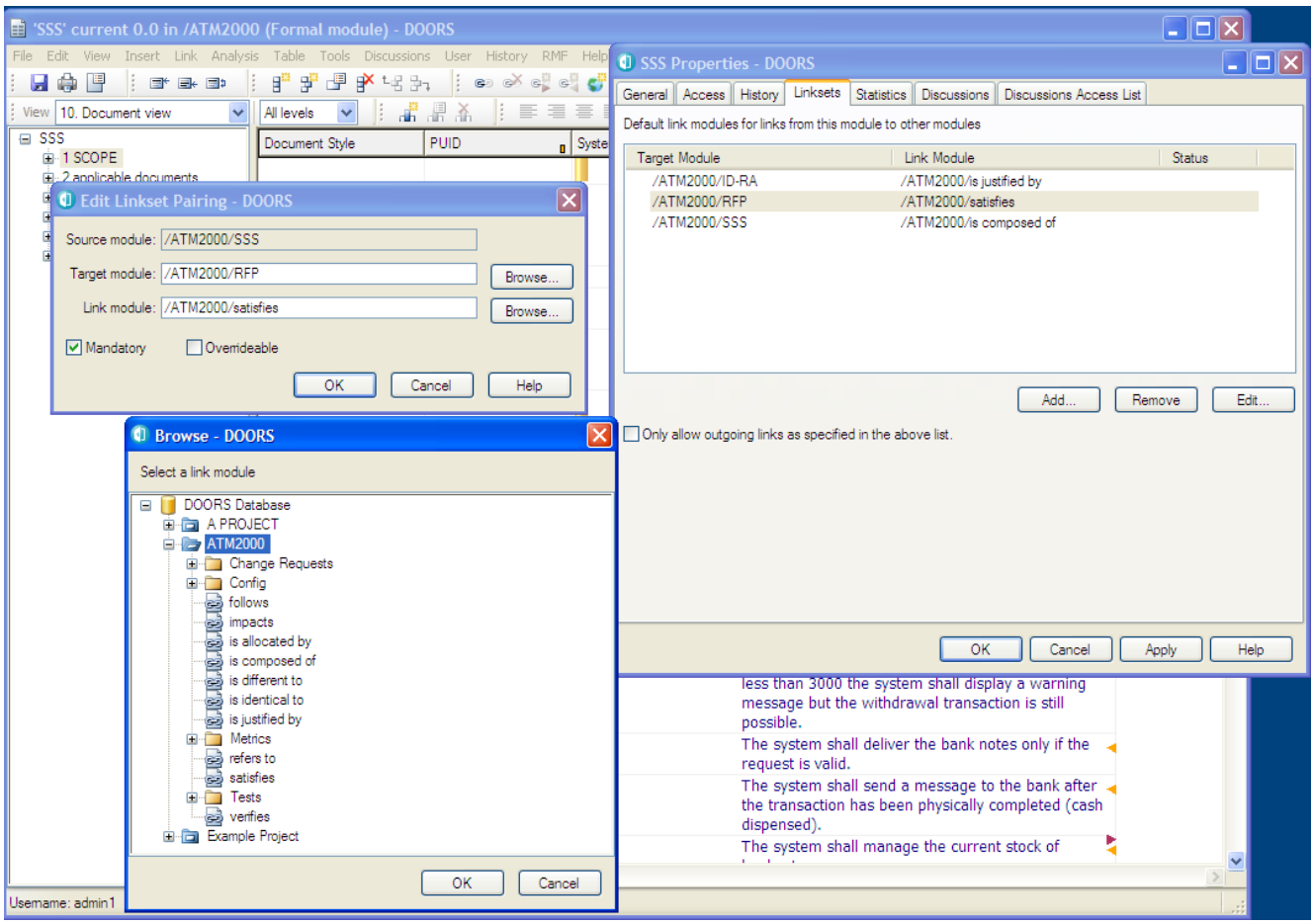


Figure 39 : linkset pairing configuration with DOORS

Why IRDRMFAO can't do it automatically ?

To do this, you need a macro-vision of your data model that IRDRMFAO can't determine alone: for example, you can have 3 ranks of SR modules and authorize "satisfies" links between rank n & n-1, and n-1 & n-2, but prevent direct link between n & n-2.

With IRDRMFAO , there is now a better support to do this operation: you may use the **Explorer** tool to display a map of your project, and to manage the dependencies between modules. The tool shows you what are the linksets compatible or not compatible with your RMF Project Model.

Example:

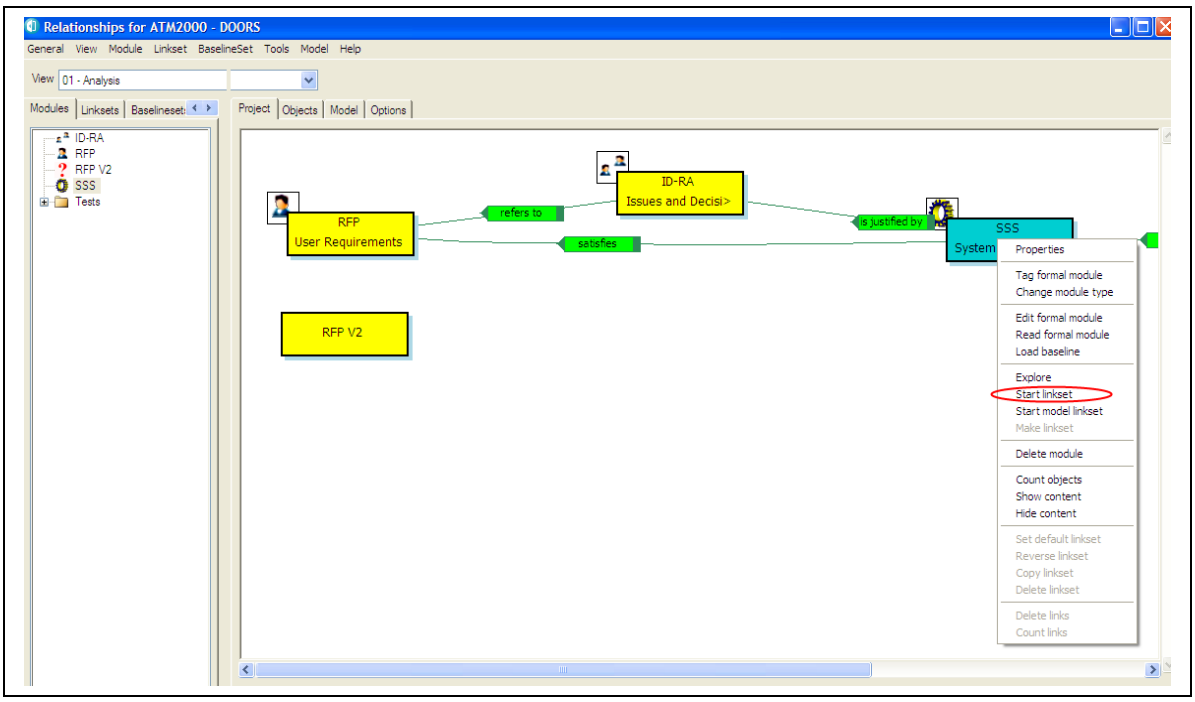


Figure 40 : Start model linkset in Explorer

You may use the operations "Start model linkset" and "Make linkset" to create a new default linkset, compatible with the RMF Project Model, between two modules. To have more details about this tool you should read the document describing the **Explorer** tool. The **Explorer** has a lot of other functionalities.

4 Requirement analysis

4.1 CHARACTERIZE REQUIREMENTS

Once your RMF objects (i.e. requirements, capabilities, configuration items,…) have been identified in your module, you have to manage them by making use of attributes. For this, all RMF module types make available at least an "Analysis" view, and sometime specific analysis views like "Critical requirement list". In addition, most of the traceability matrix views add columns with attributes for analysis. For more details on the contents of views and the attribute meanings, refer to the appendix A. To customize attributes, refer to § 7.2 ADAPTING MODULE ATTRIBUTES.

The following table summarizes the availability and intended usage of the views that you will find in RMF predefined module types.

Table 2: List of standard RMF semantic views

	View name	UR	SR	PBS	IDJ	IVV	Remarks
Document approach	Document view	X	X	X	X	X	
Generic Analysis Views	Requirements analysis	X	X				
	Configuration item analysis			X			
	IVV procedure analysis					X	
	Issues and Decisions analysis				X		
Specific Analysis Views to point out a particular aspect	Risk analysis	X	X				
	Key requirements list	X	X				
	Requirements in negociation	X	X				
	Test analysis	X	X				
Traceability Matrix Views	Associated issues Matrix	X	X	X		X	
	Compliance Matrix	X	X				
	Verification Matrix	X	X				
	IVV Matrix					X	
	Allocation Matrix		X				
	Justification Matrix		X	X		X	
	Upper requirements satisfies Matrix		X				
	Allocated requirements / test Matrix			X			
	Decisions justify Matrix				X		
	Issues refers to Matrix				X		

The templates are containing also some other views, but that are generic or "technical" views:

- **Configuration view**. Display information on the creation and modification dates and users.
- **Discussion view**. Display the opened discussions.
- **Link info**. Display all links, at level one and recursively.
- **Suspect Links**. Display suspicions on all outgoing links.
- **Rich Text Format**. Display RTF markers of the Object Heading and Object Text attributes (without OLE information).

Some RMF tools may also require specific supplementary views for their configuration. These views may be module specific, but they can also be defined into the model.

- RMF Exchange is using views to filter attributes to export and to import.
- The documentation generation and other export tools are using views to describe exported information.

4.2 DEFINE ISSUES AND DECISIONS

In order to keep track of significant issues encountered during your analysis and to record the decisions taken, it is highly recommended that the "Issues and Decisions" module (IDJ) type is used.

The IDJ module provides enriched traceability between a pair of modules, as for example between the following pairs: UR-SR, IVV-UR, IVV-SR or PBS-SR.

The IDJ module connects together only relevant RMF objects with issues, not the whole module. So it is not necessary to have links to all objects in the "incoming" module. In the same way, reference links to all objects in the "outgoing" module are not mandatory.

For significant issues with impact outside the local team (customer, subcontractor, business, major tradeoffs, etc) use the "Issue - Decision" construct in the Requirements Analysis and Design Analysis modules to capture the issue and, when resolved, the decision. It is useful to manage negotiation with customer using the "status" attribute and/or other attributes you may want to create.

Notice also that a "Rationale" object attribute field exists in the User Requirement and System Requirement modules for "routine" decisions and issues within the responsibility of individual engineers or co-located teams. It is recommended that this mechanism is used to record less significant issues and decisions.

With the version 9 of DOORS, it is also possible to use the "Discussion" concept to capture the different remarks concerning requirements. The "Issues and Decisions" model is more formal and more descriptive.

4.3 Identify RISK and key requirements

4.3.1 Introduction

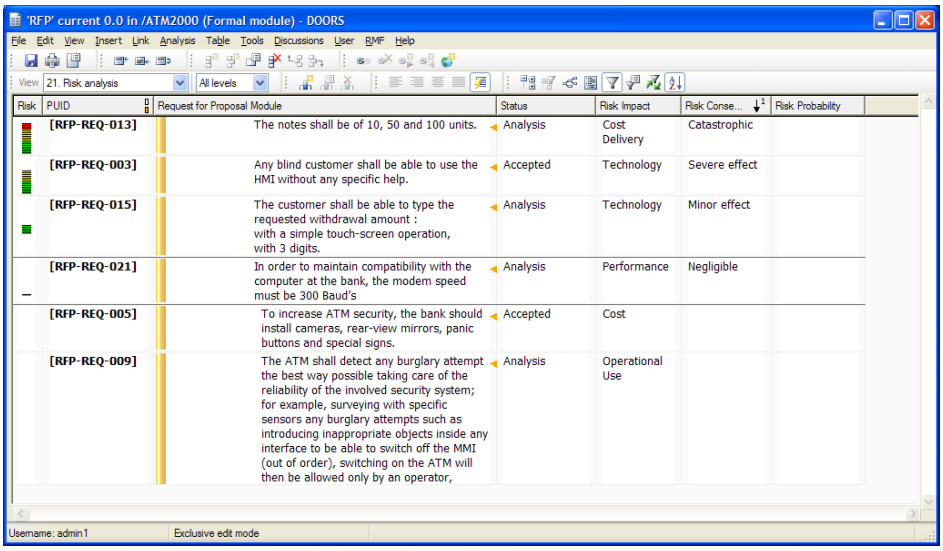
IRDRMFAO provides specific facilities to assist in the management of requirements which are particularly important to the project, either because they are risky or because they are key requirements for another reason (e.g. stage payment milestones). The former category is called "Critical requirements" and the latter are called "Key requirements", and of course some requirements can be in both categories.

4.3.2 critical requirements

During requirements analysis, you may identify some requirements as being particularly risky for the project. To lend weight to them and allow you to follow them closely, IRDRMFAO provides a dedicated view "Risk analysis".

The procedure is:

- First step, in the view "Requirements analysis", set the enumerated "risk impact" attribute. Setting this attribute both marks the requirement as risky and qualifies the type of risk (Technology, performance, cost, delivery…). Notice that the attribute can take several values.
- Second step, in the view "Risk analysis" (which is filtered on risk, see figure below), you can additionally quantify the level of risk by setting the attribute "Risk Level", and quantify the probability of the risk occurring by setting the "IE Risk Probability" attribute. A graphic bar chart allows you to quickly spot risk visually, as 'risky' requirements have a coloured border. This is particularly usefull when you unset the filter icon and see the critical requirements among all others.



4.3.3 Key requirements

During requirements analysis, you can identify some requirements as key for the project. To lend weight to them and allow you to follow them closely, IRDRMFAO provides another dedicated view "Key requirements list".

The procedure is:

- First step, in the view "Requirements analysis", for the key requirements set the boolean "Key requirement" to true.
- Second step, display the view "Key requirement list".

4.4 LINK RMF OBJECTS

Once RMF objects have been identified (Requirements, functions…), you will have to link them according to the RMF data model (See §2.2 RMF GENERIC DATA MODEL) or your own customization.

There are two steps to link objects:

- Select a link name (optional),
- Link objects.

4.4.1 LINK SET SELECTION

Most of the time, you will not need to select the link name because you will use the default linksets you had already defined (Refers to § 3.2.2 CREATE A NEW RMF Module).

If you have not define the default link set or want to use another one, select a link module which will be the default link set for all links you will make in the current session, until you select another one.

To select a link module, activate the utility by the menu "RMF ->Define default link module" from database window or the module window.

It should always be better to use only the linkset pairing definition and never the default link module, because this concept allows you to make more mistakes according to the data model.

4.4.2 LINKING RMF OBJECTS

4.4.2.1 GENERAL PRINCIPLES

RMF objects are linked with the standard way of DOORS.

Notice that if your RMF objects have a hierarchy, the incoming or outgoing links must connect the **main object (father, not children objects)**.

Caution: Pay attention to respect the directions of the data model links. Notice that in the data model the links are bottom-up oriented. This is not a coincidence, it is for traceability convenience! To avoid any mistake, you should define all linkset pairing in all modules, and set the option "Only allow outgoing links as specified in the above list" to ON.

Example:

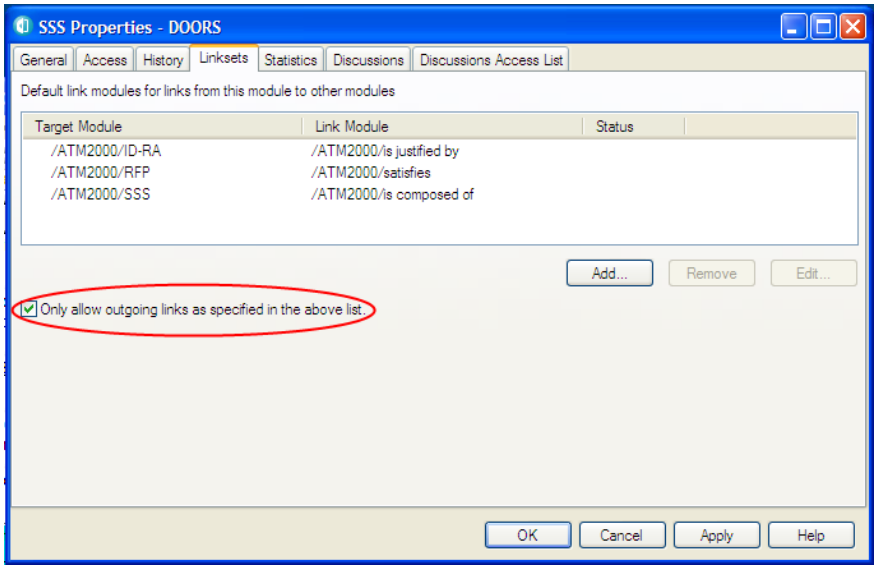


Figure 41 : Option "Only allow …" in Linksets tab

Badly oriented links can be corrected with the tool "*Explore -> Project*" (see §4.4.3), from the RMF menu in the DOORS Database window. This tool will point out the linksets that are not compatible with your RMF Project Model.

As you put links inside your project, you will be quickly able to see the effect by displaying the predefined traceability views. Refer to the table §4.1 "*List of standard* RMF *views*" to quickly locate the right view according to module type and link module.

4.4.2.2 REQUIREMENTS SATISFY UPPER LEVEL REQUIREMENTS

Typically, System Requirements (template SR) are satisfying User Requirements (template UR). This is the typical use of the link module "satisfies", but the data model also allows a cascade of modules derived from the "S" template. For example sub-system requirements satisfy system requirements. To allow dependencies "System Requirements" to "System Requirements" you should modify the default model.

4.4.2.3 REQUIREMENTS / FUNCTIONS ARE ALLOCATED TO CONFIGURATION ITEMS

System Requirements are allocated to Functions and Functions are allocated to Configuration Items, both using the link module "is allocated by". This is the correct methodology. IRDRMFAO also allows Requirements to be allocated directly Configuration Items, again by using the "is allocated by" link module.

4.4.2.4 LINK RMF OBJECTS WITHIN A SAME MODULE

It is possible to link objects within a same module. One use of such links is to model an additional hierarchy, compared to the hierarchy modelled by standard DOORS headings and object structure, for example if the DOORS hierarchy is used to model a functional decomposition, explicit links could be used to model a physical hierarchy.

Another use is to mimic the implicit DOORS hierarchy with an explicit one created from links, because this greatly facilitates the creation of traceability or impact views, without DXL development. For this situation, IRDRMFAO provides a utility which will automate the process.

For example, if your document structure in DOORS, reflects the decomposition between Capabilities and Requirements, it is possible to automatically generate explicit links between them.

4.4.3 CHECK THAT YOUR PROJECT LINKS CONFORM WITH DATA MODEL

Users working with DOORS may have several DOORS *Link Modules*, either from the RMF model or defined by the users.

The creation of links using for example drag and drop mouse operation will sometimes result in inconsistencies such as being in the wrong direction or being created in the wrong link module.

No automatic consistency checks are provided by DOORS and the only way to ensure the links are compliant with the Project Model is to do check by hand or to use a specific tool. Within IRDRMFAO, this functionality is supported by the **Explorer** tool. This tool allows you to examine your project from the dependencies point of view.

Example:

In the example, you can see that the selected linkset (from "SSS" to "RFP V2") is not compatible with the model because of three displayed information:

- The arrow containing the name of the link module is "opened" (i.e. no square at the opposite extremity of the arrow)
- The linkset has the symbol ? in the link module/linkset explorer

- When using the operation “Properties”, the field “Model” has the value “No”

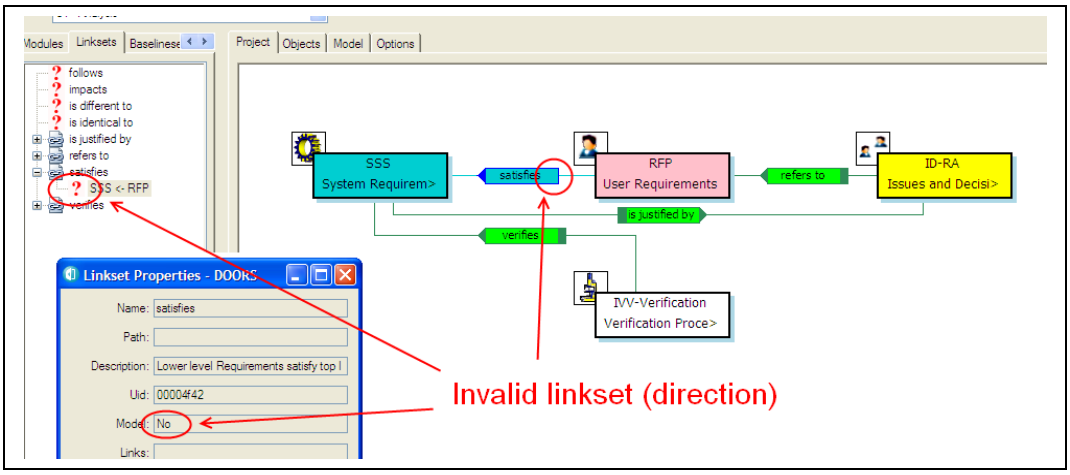


Figure 42 : Invalid linkset in Explorer

The tool offers you a set of operations allowing managing the linksets in order to make your project consistent with the RMF Project Model:

- Reverse the direction of the links/linkset
- Copying the links/linkset into a different link module
- Deleting the links/linkset
- …

The tool itself is not able to repair automatically your mistakes, it is your responsibility to analyze the mistakes (if any), and to make the correction. Frequently, the problem will not be a mistake but only a missing declaration into the model. In this case you have to complete or repair the modeL

Refer to the **Explorer** manual for more details on this tool.

4.5 CHECK DATA CONSISTENCY

When you have identified and linked your requirements, you need to check and validate your data, to detect and correct errors.

This verification may be manual, only by reading the information saved in DOORS, but before this manual task you may detect a lot of errors by using at least one of the two functionalities provided by IRDRMFAO.

- The check project or module consistency tool is able to check a set of some predefined rules.
- The functionality Integrity Check has been designed to be flexible and opened, allowing the users of IRDRMFAO to define their own integrity rules, specific of the process or of the data model deployed, in order to complement the generic rules already provided by IRDRMFAO.

The two functionalities are different and not compatible.

4.5.1 CHECK PROJECT AND MODULE CONSISTENCY TOOL

This tool should be used to verify that:

- The unicity of the requirement identifiers (the “IE PUID” attribute)

- The model links are only between RMF objects
- The links associated with the RCM objects in the Working or Deleted state.

This tool may be executed at project level or at module level. When executed at project level, all the modules are scanned, and the unicity of a requirement identifier is verified in all the project. At module level, the unicity is checked only in the scope of the module.

To check the consistency of a module, calls the "Consistency check" operation. It opens a dialog box such as:

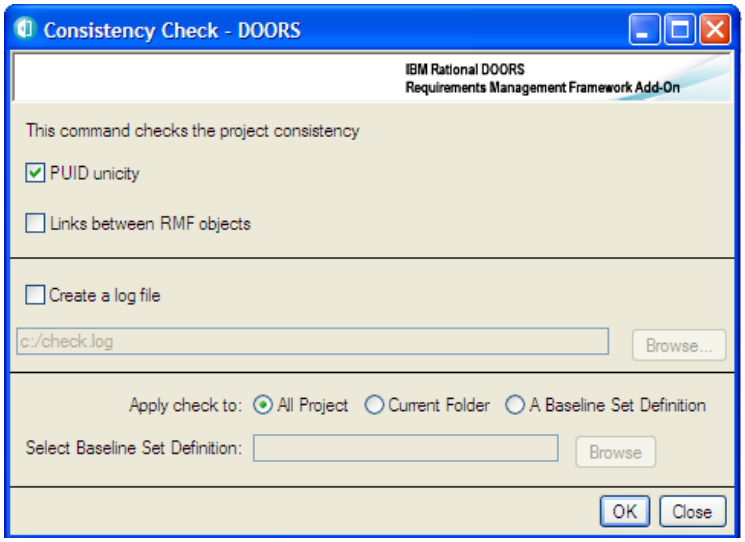


Figure 43 : Consistency check (no RCM configured)

The corresponding check rules are:

- **PUID unicity**: check that each RMF object in the selected scope as a unique RMF identifier (the prefix of each module must be unique)
- **Links between RMF objects**: check that the semantic links (i.e. the relationships defined in the project model) are only between RMF objects. A RMF object is an object with the an "IE Object Type" attribute value non empty.

If the RCM management is activated for the current project, the dialog box will be such as:

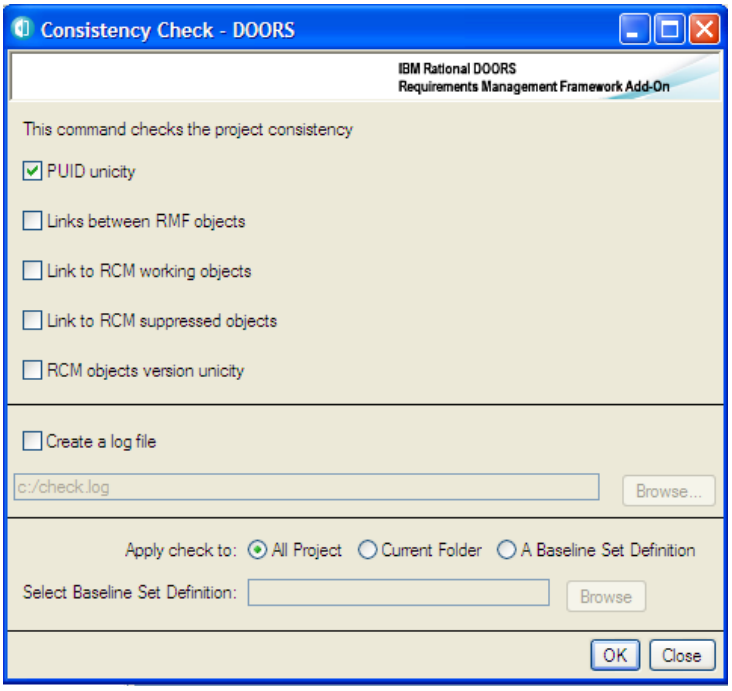


Figure 44 : Consistency check (with RCM configured)

The complementary check rules are:

- **Links to RCM working objects**: check that they are no links to objects with the **Working** RCM status in modules under RCM control .
- **Links to RCM obsolete objects**: check that they are no links to objects with the **Obsolete** RCM status in modules under RCM control .
- **RCM objects version unicity**: check that if there is a link to an object in a module under RCM control, they are no links to another object of the same version graph (for example links to the version 1 and to the version 2 of a requirement).

To have more details about the RCM behaviour you should look at the RCM manual.

The next option allows the definition of a Log file, to memorize the rules violations detected by the consistency check.

The last option is the definition of the part of the project in which the consistency check should be executed. It is possible to select:

- The all project
- A sub-folder inside the project
- A baseline set definition, i.e. a set of modules in the project

After having defined the options, click on the OK button. You get a dialog such as:

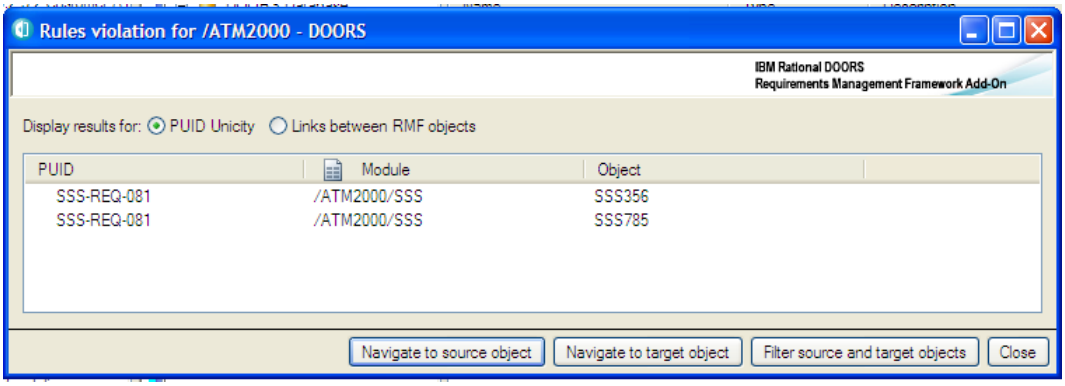


Figure 45 : Violated rules for "PUID unicity"

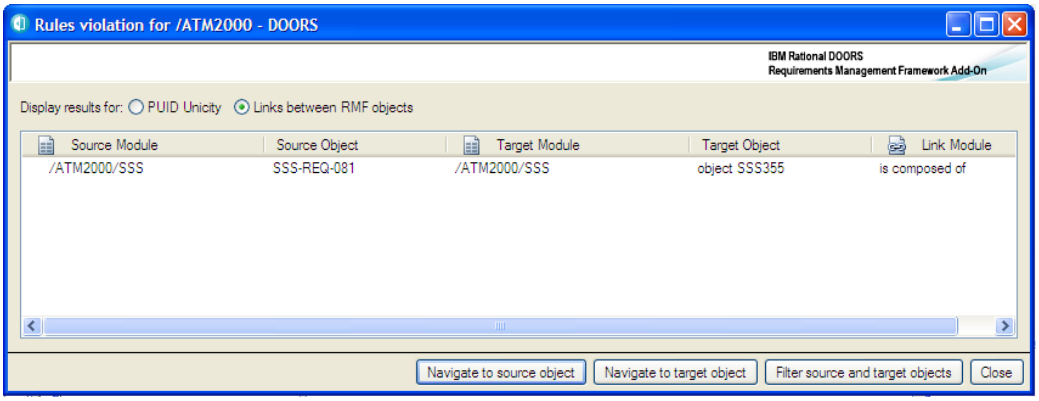


Figure 46 : Violated rules for links

The result list displays the violated rules. To see the different types of violations you must select the rule that you want to examine. The information displayed is not the same, if the violated rule is associated to an object ("PUID unicity") or to a link (all other rules).

A violation associated with an object contains only the reference of the object (source module and object reference), a violation for a link contains the description of the source and target modules and objects, and also the link module.

The object is described by the RMF identifier if any, or by the DOORS identifier. For an object under RCM control, the version and the RCM status of the object is added to the PUID.

Example of log file:

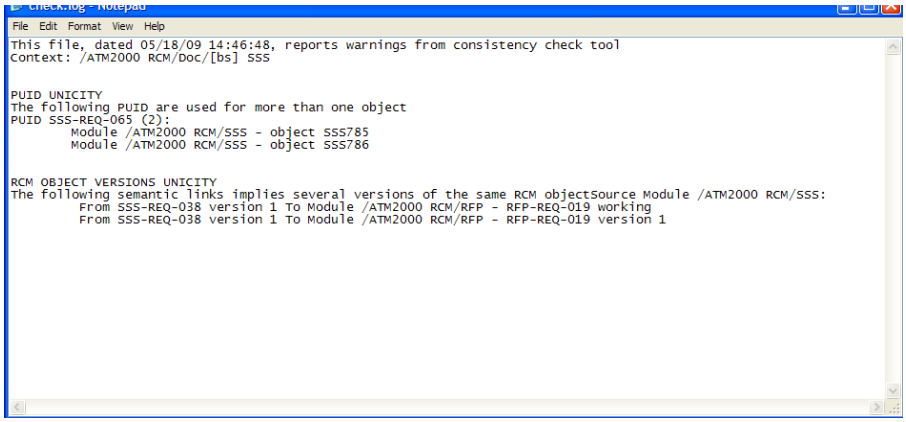


Figure 47 : consistency check log file

To examine the violation you can use the different buttons or execute a double-click on a rule violation. The double-click is equivalent to the "Navigate to object" operation.

The available operations are:

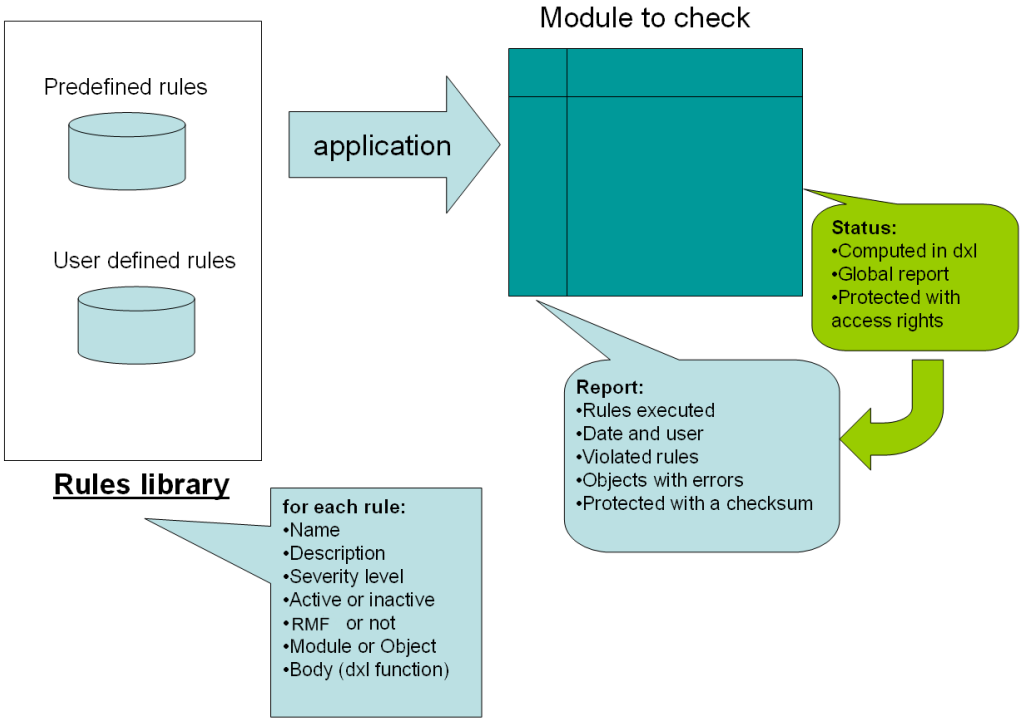
- **Navigate to source object:** open the source module and the source object corresponding to the selected violation
- **Navigate to target object :** open the target module and the target object corresponding to the selected violation
- **Filter source and target objects:** open all source and target modules and filter the objects listed in the violations

You may execute the tool from a module, in this case you do not have the "Apply check" and "Baseline set" fields defined in the dialog box. The scope of the check is implicitely the current module.

4.5.2 INTEGRITY CHECK

The consistency check tool is limited because no additional rules are possible. The Integrity Check has been designed to be flexible and opened, in order to be able to associate specific integrity rules to any model.

Architecture of the Integrity Check:



The rules are defined by coding some DXL functions with an interface like:

```
bool rmf_object_puid_syntax_(Module,Object,DxlObject)
```

The first parameter is a module to check, the second may be null if the rule is at module level, or it is the current object is the rule is at object level. The third parameter is a DxlObject allowing to store information during the execution of the rule. The result is TRUE if no violation is detected, and FALSE if a violation is detected.

The rules are "linked" with the Integrity Check mechanism, and associated with some information allowing to manage the rule. The different information associated with the rules are:

- The name of the rule. For example "R 1.1.1".
- The description of the rule. For example "The PUID of a RMF Object must be non empty and compatible with the project or module configuration".
- The level of the rule: it can be module level or object level.
- The RMF level of the rule: it can be RMF , i.e. only a RMF module or a RMF object should be verified, or not RMF. In this last case any module or object may be verified.
- The severity level. It can be a critical violation (Error) or a non critical violation (Warning).
- The activation status. If the rule is inactive, it can not be seen in the configuration or applied to a module.

The name associated with a rule should be unique for the set of rules (predefined AND user defined).

This information is already defined for the predefined rules, but it is possible to change these values, the corresponding DXL file is not encrypted. The code itself is not public. To modify the behaviour of a predefined rule, you have to inactive it or not to "link" it, and to write a new DXL function to replace it.

At execution time, the set of active rules is applied on the module to verify. The verification of one module is independent of the verification of the other ones: it is not possible for example to check the unicity of the PUID on a set of modules with this mechanism.

When executed on a module, two module attributes will be created if required, and initialized.

- **IE Check Report** is a Text attribute. It contains the execution report in a compact format.

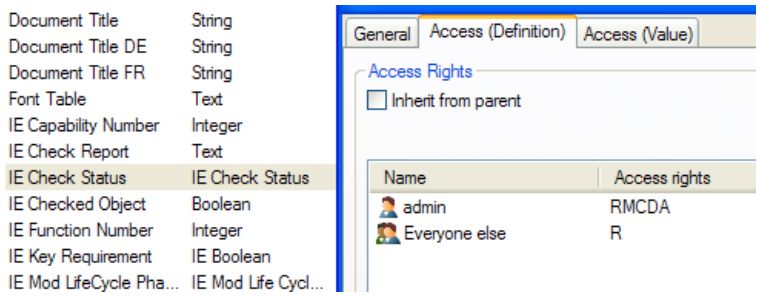
Example:

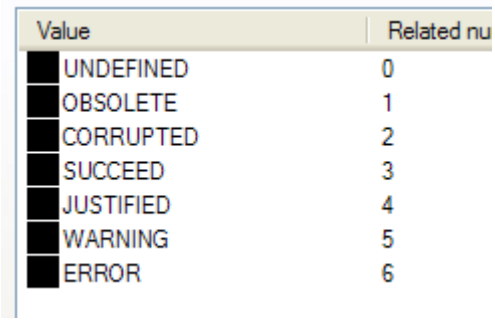
```
[Warnings]
R 2.2=132,359,668
[Errors]
[LastVerification]
Author=admin
Date=1232552174
Elapse=1
Rules=R 1.1.1,R 1.1.2,R 1.1.3,R 1.1.4,R 1.2.1,R 1.3.1,R 1.3.2,R 2.1,R 2.2,R 3.1
[Checksum]
Date=1232552174
Value=1303303858
```

The information contained is:

  - The list of executed rules
  - The date and time of the last execution, with the user and the elapse time
  - The violated rules, and the objects on which the rules are violated
  - The checksum value

- **IE Check Status** is a DXL enumerated attribute. The code of the attribute is located into IRDRMFAO code and encrypted. The goal is to protect the attribute from any manual modification. The attribute definition is protected with access rights.





The algorithm implemented in the DXL status attribute has been defined to be able to detect any corruption or modification. The different values are:

- **UNDEFINED**: no report yet. The rules have not been executed.
- **OBSOLETE**: some modification has been done into the module since the last verification. This status is based on the history. A modification not recorded into the history will not be taken into account.
- **CORRUPTED**: the report checksum is invalid or the format of the report is wrong.
- **SUCCEED**: no integrity violation has been detected.
- **JUSTIFIED**: some integrity violation has been detected, but a Check Manager has "accepted" the report.
- **WARNING**: at least one not critical integrity violation has been detected.
- **ERROR**: at least one critical integrity violation has been detected.

Several possibilities are provided by IRDRMFAO to apply the integrity rules on a given module. You may execute the rules through triggers. In this case the trigger is a module close trigger, applied only if the module has been opened in Visible Edit mode. The trigger must be defined locally for some modules. It is also possible to execute the check directly on user decision, by calling an operation from the graphic user interface. You can apply the check on the current module; it can be also on a set of selected modules.

In any case, you must first define the integrity check configuration at project level to be able to use any of the integrity check functions.

4.5.2.1 Integrity check at project level

You may execute the operation "Manage Integrity Check". Only a "Check Manager" may use this operation. You should use it only on small size projects, if the majority of users are not well trained.

A dialog box opens:

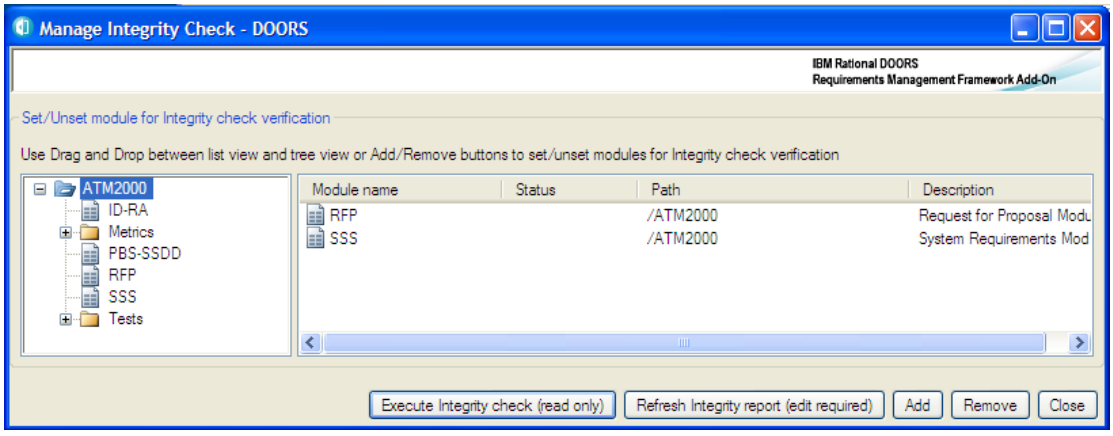


Figure 48 : project integrity check

To check the integrity status of a set of modules, you must drag the modules from the tree view to the list view, select one or several modules in the list view and then you may click the button "Execute Integrity check (read only)" or the button "Refresh integrity report (edit required)".

You may drag a folder or the all project to initialize the list view with all contained modules. You may also select an item in the tree view and click to the "Add" button.

To clear the list view, you may also drag the modules from the list to the tree view, or select one or several list elements and click the "Remove" button.

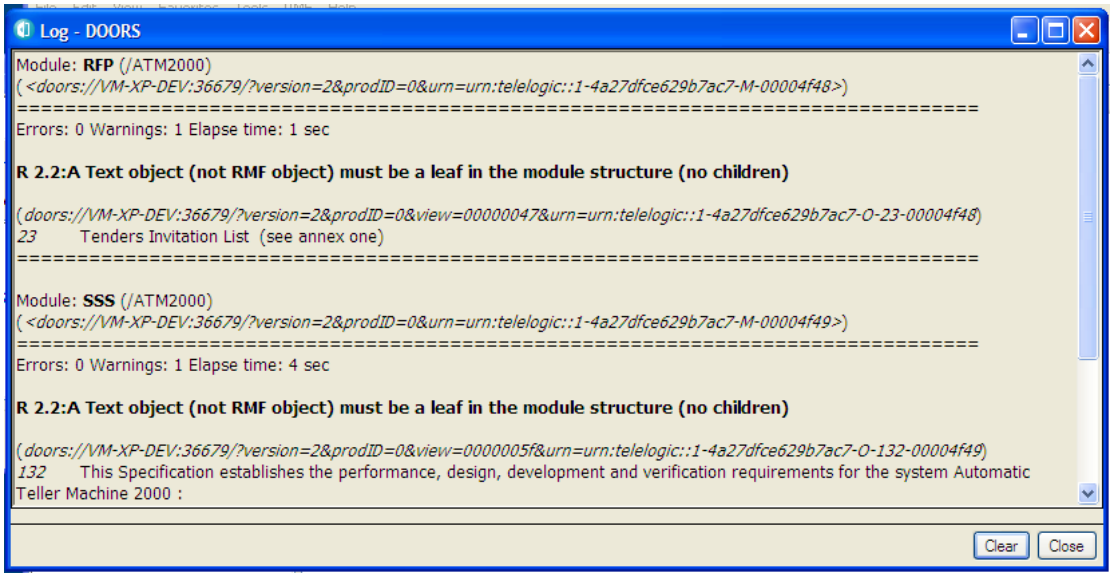


Figure 49 : project integrity log

Each detected violation is written into the Log window. This window may be saved into a text file or printed. If you use the "read mode", no attribute is created into the checked module.

The operation "Execute Integrity check" doesn't create any report or status attribute into the checked modules that are opened in read mode. The report is in the log window and the status is in a column of the list view.

The operation "Refresh Integrity report" creates and initializes the report and the status attribute into the checked modules that are opened in edit mode. The report is also in the log window and the status is also in a column of the list view.

4.5.2.2 Integrity check at module level

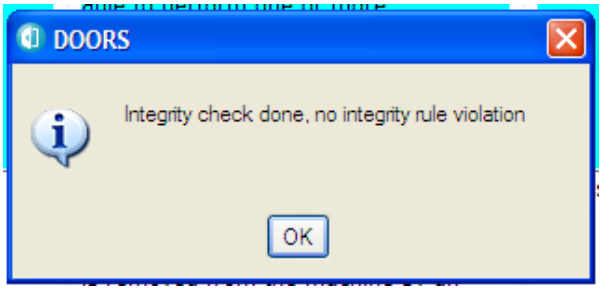
Different operations are possible at module level. The two main operations are "Integrity Check" → "Execute Rules" and "Integrity Check" → "Display Report".

"Execute Rules" should be used to apply the configured integrity rules for the module. This list of rules to execute may be defined from the project configuration or from the module configuration if it has been defined locally.

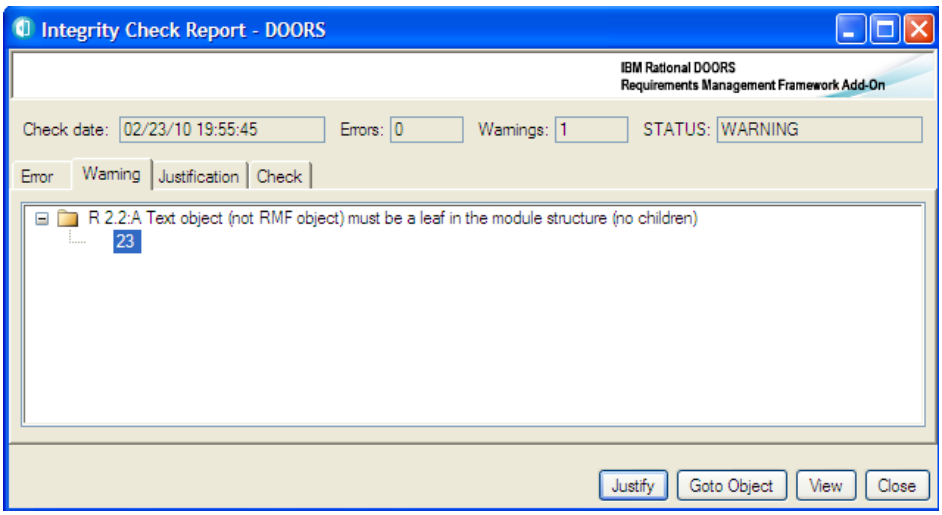
The rules may be executed in Edit mode or in Read mode, and even in a baseline of a module. The report and status attributes are created or refreshed only in Edit mode.

At the end of the execution:

- If no violation is detected, a message is displayed



- If at least one violation is detected, the report display tool is opened:



In Edit mode, the result of the verification is saved into the report module attribute, and it is possible to consult it even after having closed the session, by using the operation "Integrity Check" → "Display Report". In Read mode, the report is not saved and you will have ti execute again the check to find the violations.

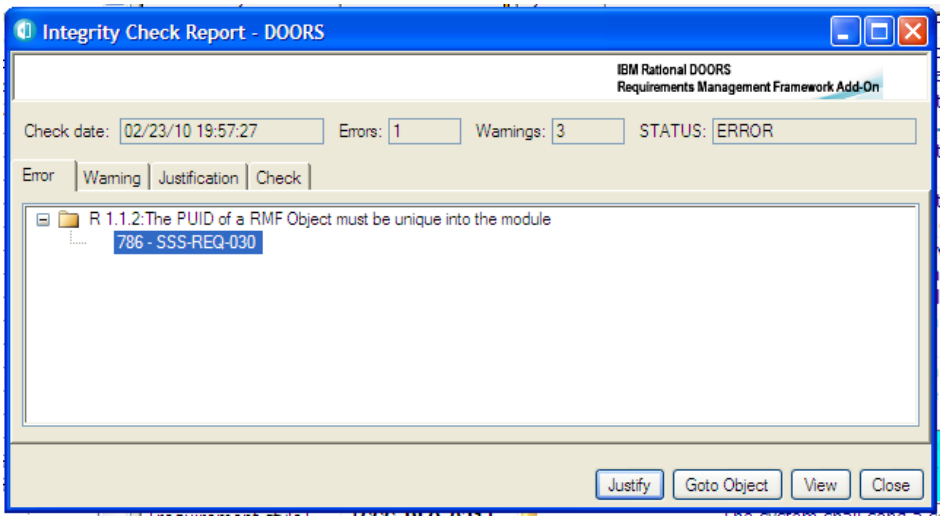
You can also use the operations "Integrity Check" → "Create trigger" and Integrity Check" → "Delete trigger" to manage the integrity check trigger of the module. It is possible only in Edit mode.

The "Create trigger" operation defines a module close trigger that will apply the integrity rules each time the module is closed after an edit session in visible mode.

4.5.2.3 Integrity check report dialog

This dialog is dedicated to the analysis of the integrity check report information. The information found into the report attribute (or from another source in case of read mode) is compared with the configuration and displayed in a more readable form.

Example of display:



The meaning of the fields in the top of the window is:

- **Check date**: contains the date of the last execution of the integrity check.
- **Errors**: number of critical integrity rules violated
- **Warnings**: number of not critical integrity rules violated
- **STATUS**: global integrity status of the module (identical to the status module attribute)

The different tabs contain the information required to execute a fine grain analysis of the problems.

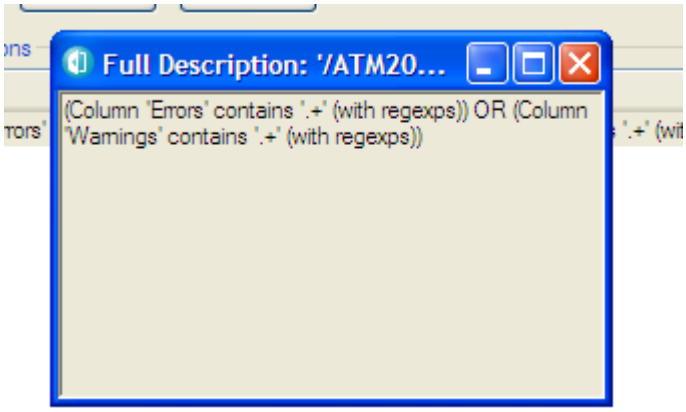
- **Error**: contain the list of violated critical rules, and the objects violating the rules for object level rules

This tab and the **Warning** tab may be used to find easily the wrong objects. You can click on one object visible below a violated rule, or click the "Goto object" button. The corresponding object is selected into the module.

You may also click the "View" button. The current view of the module is modified to display all anomalies into the module:

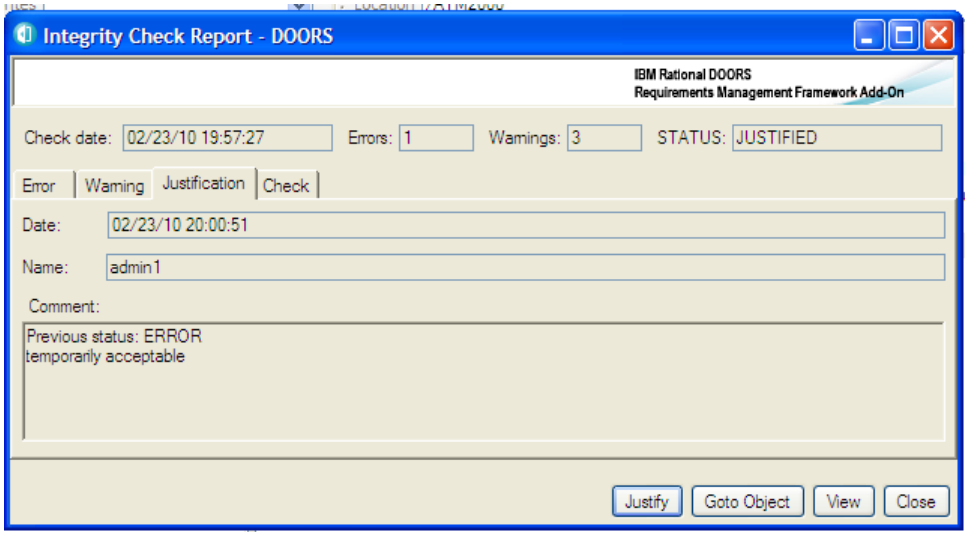
Two DXL columns are added before the main column, to display the rules violated by each object, one column for critical rules and another for non critical rules. These columns are DXL columns pointing to the IRDRMFAO code.

A filter is also created:

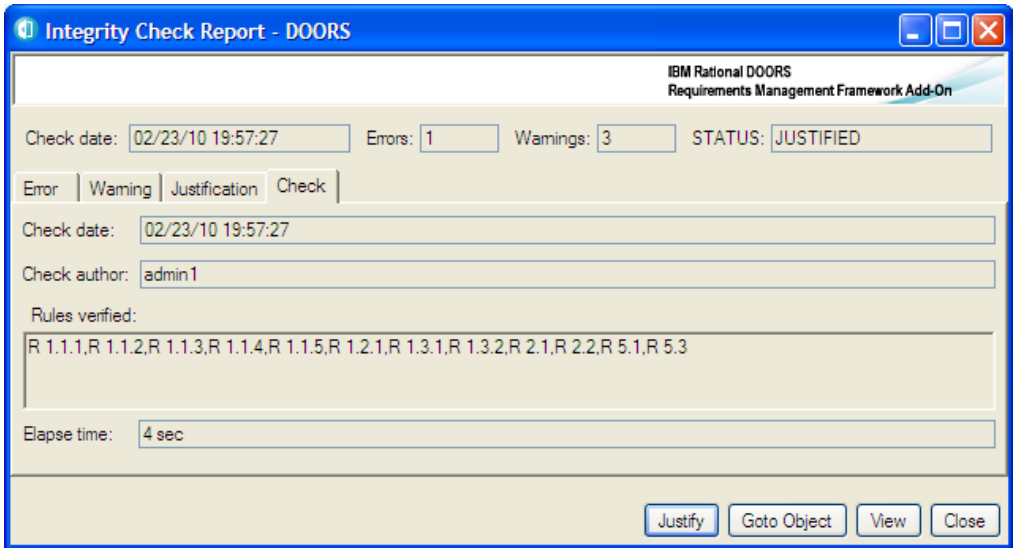


The columns and the filter may be saved into a view. They will be automatically refreshed in case of modification of the integrity check report embedded into the module.

- **Warning**: contain the list of violated not critical rules, and the objects violating the rules for object level rules
- **Justification**: if a Check Manager has accepted the report, it contains the information required to validate the acceptance. By default this tab does not contain any information. It is not empty only if the integrity check report has been accepted, even in case of error, by clicking the button "Justify". The definition of a "comment" is mandatory.



- **Check:** contains some useful information about the last integrity check (check date, user, rules in the configuration , elapse time of the check)

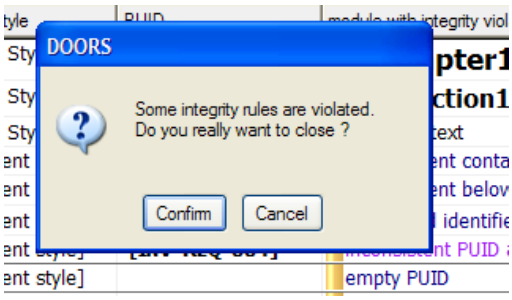


4.5.2.4 Integrity check in triggers

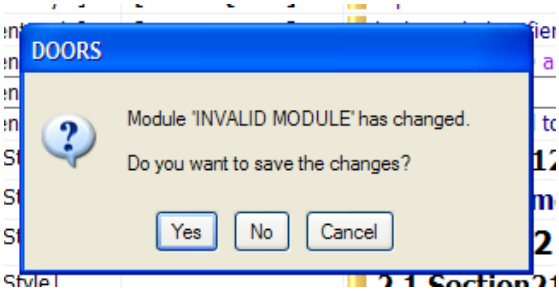
If an integrity check trigger has been defined for a module, at project level or locally at module level, it is a module close trigger that will be executed:

- If all configuration information is consistent
- If the module is in Edit mode and if it is visible
- If the check status is not JUSTIFIED

In case of detection of an error, a confirm message is displayed to prevent from closing the module:



In general, even in case of save before the execution of the close, the execution of the trigger modifies the data inside the module, and you need to save again the module:



If you choose "Cancel" for the first message, the module will not be closed and the check report display box is automatically opened to allow you to repair the errors.

4.5.2.5 Predefined integrity rules

IRDRMFAO contains a set of predefined integrity rules. These rules are generic, and applicable to any project, even if the data model is different from the default RMF model. This set of rules will be extended in the next versions of IRDRMFAO to cover the maximum number of errors.

Name	Description	Level	RMF Level	Criticality
R 1.1.1	The PUID of a RMF Object must be non empty and compatible with the project or module configuration	Object	IRDRMFAO	Error
R 1.1.2	The PUID of a RMF Object must be unique into the module	Object	IRDRMFAO	Error
R 1.1.3	The PUID of a RMF Object must be identical compared to the last Major baseline	Object	IRDRMFAO	Error
R 1.1.4	The Object Type of a RMF Object must be identical compared to the last Major baseline	Object	IRDRMFAO	Error
R 1.1.5	The PUID counter of a RMF Object must be inferior or equal to the corresponding global counter	Object	IRDRMFAO	Error
R 1.2.1	The hierarchical structure of a RMF Object must be compatible with the project or module configuration	Object	IRDRMFAO	Error
R 1.3.1	The origin of a RMF semantic link must be a RMF object	Object	DOORS	Error
R 1.3.2	The destination of a RMF semantic link must be a RMF object	Object	DOORS	Error
R 2.1	The Heading and Text attributes must never be defined together	Object	DOORS	Error
R 2.2	A Text object (not RMF object) must be a leaf in the module structure (no children)	Object	DOORS	Warning
R 3.1	The prefix attribute of a module must not be empty	Module	DOORS	Warning

If you want to modify the parameters associated with the different rules, you have to open the DXL source file:

[IRDRMFAO DIR]/lib/dxl/addins/irdrmfao/check/predefdcl.inc

The file contains a sequence of declarations like :

```
{
    int idx = CheckRuleDeclare(
        "R 1.1.1",
        "The PUID of a RMF Object must be non empty and
compatible with the project or module configuration",
        false,
```

```
            true,
            true,
            true)
      if(idx != -1)
            put(CHECK_RULE_TABLE,
                  rmf_object_puid_syntax_
                  , idx, CHECK_RULE_FUNCTION_FIELD)
}
```

You may modify the different parameters of the call to the function « CheckRuleDeclare », but do not modify the « put » instruction.

The order of the parameters is :

- Name
- Description
- Module or Object level (true is Module level and false is Object level)
- RMF or not RMF (true is IRDRMFAO level and false is DOORS level)
- Criticality (true is Error and false is Warning)
- Activation (true is active and false is inactive)

4.5.2.6 User defined integrity rules

You may replace predefined rules or complete the set of rules by inserting your own rules into the DXL source file:

[IRDRMFAO DIR]/lib/dxl/addins/irdrmfao/check/userdef.inc

This file contains already four examples of rules:

- R 5.1. This rule checks that an object is not empty, i.e. that the Object Heading or the Object Text attributes have a value. The rule is an object level, IRDRMFAO level, not critical rule.
- R 5.2. This rule checks that the module attribute "ModStatus" is not empty if it is defined into any module (RMF and not RMF). The rule is a module level, not IRDRMFAO level, critical rule.
- R 5.3. This rule checks that the object attribute "ReqStatus" is not empty for a RMF object, if it is defined into a RMF module. The rule is an object level, IRDRMFAO level, critical rule.
- R 5.4. This rule checks that a RMF object has no children. The rule is an object level, IRDRMFAO level, not critical rule.

The declaration of the rules 5.2 and 5.3 are commented into the code, and not visible at GUI level.

The interface of all rules is identical:

```
bool <function_name> (
      Module curMod,
      Object curObj,
      DxlObject context
      )
```

The call of the function by the integrity check mechanism depends on the declarations associated with the function:

- **Module level**

  The *curObj* and *context* parameters are not used. The function is called only once for a module.

  - **RMF level**

    The function is called only for a RMF module, i.e. a module with the module attribute "IE Mod Type" not empty.

  - **Not RMF level**

    The function is called for any module.

- **Object level**

  All parameters are used. The parameter *context* is always defined. The function is called a first time with *curMod* defined and *curObj* null. The goal is to give the ability to the function developer to initialize some data that are saved into the DxlObject object, for example a skip list to memorize a list of values. Then the function is called on the different objects to check. Finally, the function is called with *curMod* null. The goal is to give the ability to the function developer to release the data saved into the DxlObject during initialization or function execution. The DxlObject object itself must not be deleted. You can also test the execution of the constructor by using the "init" field of the DxlObject. It is defined to "false" by IRDRMFAO before calling the constructor, and to "true" after.

  - **RMF level**

    The function is called on non deleted RMF objects (only the root in case of a composite RMF object).

  - **Not RMF level**

    The function is called on all non deleted objects.

When developing new rules, you must pay attention to their robustness and efficiency, because it can be executed many times and on any module of your project, specifically if you decide to check integrity rules with triggers at module or project level.

5 Defining the IVV solution

5.1 Introduction

Each RMF object identified as a "Requirement" in UR/SR modules is a "Reference Requirement" and should be formally tested. There is no point in defining a requirement, without any associated test. The test is described as an Integration, Verification or Validation test according to its level (Refer to SYS-EM methodology), in summary the terms are applied as follows:

- User Requirements are proven by Validation testing,
- System Requirements are proven by Verification testing, and
- Design Requirements/Specifications are proven Integration testing

Each test should be defined in terms of (1) its method, (2) who the approval authority will be and (3) where (i.e. at what level) the test will be performed. The collection of these three definitions for a requirement is termed its "IVV solution". In addition, the IVV solution usually contains some form of textual description of the scope of the testing, which acts as a constraint on customer aspirations for unnecessarily rigorous test regimes.

The RMF object that holds the IVV solution is called a test procedure. These procedures are typically located and identified in modules of the IVV type, and are linked to requirements with a "Verifies" link.

This means that each "Reference Requirement" must be reached by at least one "Verifies" link from an IVV module. The end objective is to ensure that the customer and contractor are both satisfied that the IVV solution defined is adequate to prove the requirement, and therefore a good basis for progressive project sign-off.

The standard RMF model reflects these three levels of IVV solution, but uses the same generic module type "IVV" for initializing any level.

The following sections describe how the various aspects of the IVV solution should be defined and controlled, and how RMF attributes and views can assist. The detailed definition of the attributes, relationships and views of the IVV module type are given in Appendix A Section 5.

5.2 IVV OBJECT ATTRIBUTES

5.2.1 IE Object type

The default setting of this attribute is "IVV Procedure".

Suggestion: Small projects may find it convenient to customize their IVV module by introducing three different object types for Integration, Verification and Validation Procedures; rather than having three separate modules, one of each module type.

5.2.2 IE PUID

This attribute is the unique RMF object identifier, generated automatically by IRDRMFAO. Identifiers are not normally re-used, unless the utility "Renumber objects PUID" is invoked. The structure is detailed in Appendix A Section 5.

5.2.3 IE IVV Type

This attribute defines whether the IVV procedure applies to design or production testing, or both.

5.2.4 IE IVV Test Method

A test method (Inspection, Analysis, Demonstration or Test) is associated with each procedure.

This information is accessible by means of the "IVV matrix" view within any module type (not only in IVV module type).

Comments: The method information (IADT) is located in IVV module for a methodology reason: test choices are very important, and bad choice could be catastrophic in term of cost. So, this way obliges the system engineer to create a "test procedure" object before setting the method attribute and he is thus forced to consider the testability whilst defining the requirement. The alternative (which is not prohibited in IRDRMFAO) is to create an attribute "method" in SR module but with the risk that it is either never set or remains at its default value. The recommendation to enter a scoping constraint in the test procedure object reinforces the need to consider testability as the requirement is phrased.

5.2.5 IE IVV APPROVAL LEVEL

The objective is to persuade the customer that only specific key requirement testing need to be approved by the customer, and that the tests for many requirements may be approved by the prime contractor or sub-contractors under their enterprise level QA accreditation.

Approval by the customer implies the need for approval of each test specification, each test execution and each test result document. This not only causes much cost to the customer but also adds risk to the programme as more of the customer's activities get onto the critical path and out of the control of the Project Manager.

The delivered IRDRMFAO offers an enumerated list of four levels, but projects should customize this as appropriate.

5.2.6 IE IVV responsible

This attribute is used to record the name of the person responsible for the IVV procedure, who will usually be from the test or engineering departments.

5.2.7 IE IVV SKILLS

This attribute is used to list any specific skills that are necessary to conduct the tests called up by the procedure.

Suggestion: projects may wish to make this an enumerated field so that filters for specific skill requirements may be set up.

5.2.8 IE IVV EVENT

The third important aspect of the proposed test solution is where or at what event the test is proposed to take place. This could range from a design-proving test on an equipment or software package at a subcontractor's premises through to a system user trial conducted on the whole system by the customer's end user.

IRDRMFAO as delivered sets this attribute as a "string". It is recommended that projects customize it to an enumerated list, which should be tailored to suit the particular project being supported. For example, the complete list of factory, harbour and sea trials planned for a naval project could be made an enumerated list type.

5.2.9 IE IVV EVENT PROVIDER

This attribute defines which party to the contract is responsible for planning, providing and/or conducting the IVV Event.

IRDRMFAO as delivered provides an enumerated list of "Customer", "Prime Contractor" and "Sub-contractor". Projects should customize this enumerated type to define, at least, the names of the sub-contractors.

5.2.10 IE IVV Non Regression

This attribute records whether it is unlikely that a test procedure would need to be repeated in the event of a future change to the system. It is a binary; set to "True" if the procedure is unlikely to be part of a regression test.

5.2.11 IE IVV Test SCOPE

Documenting and agreeing the objectives and scope of a test are very important; often the customer and contractor have very different views on the extent of testing necessary. For example, if the operating temperature range and the vibration performance are specified in separate requirements, the contractor might assume that vibration tests need only be conducted at standard temperatures whereas the customer might insist on vibration tests across the whole temperature range. The scoping statement would document the agreement on this, and in effect, it becomes an extension of the requirements.

Another example might be where the customer aspiration is for an actual network test with a large number of nodes, whereas the contractor anticipated a small network test and an analysis to justify the performance of the larger network.

5.3 IVV RELATIONSHIPS

5.3.1 JUSTIFICATION

The "Justification" relationship is used to link the procedure to some rationale in an Issue/Decision module that explains why any attribute of the IVV object has been set in the way it has. Typical usage might be to record why the customer felt it important to conduct vibration testing at temperature extremes.

Initially the relationship would point to an outstanding issue, when the issue is resolved and the resolution is recorded as a decision, the relationship becomes one of "Justification".

According to the standard RMF data model, this relationship is only used for issues with requirements, but projects may customize this model.

5.3.2 UNDER ISSUE PROCESS

The "Under Issue Process" relationship is used to link the procedure to a problem statement in an Issue/Decision module, from which the progress of the Issue/Decision process can be determined. The usual situation is that some attribute of the IVV object cannot be completed until the issue is resolved.

According to the standard RMF data model, this relationship is only used for issues with test support equipment, but projects may customize this model.

5.3.3 VERIFICATION

The "Verification" relationship is used to link the procedure to one or requirements that it verifies, in a Requirement module.

5.3.4 ALLOCATION

The "Allocation" relationship is used to link the procedure to Test Equipment that is necessary in order to execute the procedure.

5.4 IVV VIEWS

5.4.1 STANDARD VIEW

This view is the DOORS standard default view and shows the DOORS unique Object Identifier and the Object Heading/Text, only.

5.4.2 IVV ASSOCIATED ISSUES VIEW

This view is used when it is required to determine whether any issues have been raised about any IVV procedures. It directly provides the identity and description of any issues raised; it is necessary to follow the DOORS link to determine the status and decision of each issue.

Because the view is derived from traceability via incoming "refers to" links, it is most useful when analyzing issues that may have been raised over the allocation of test equipment, for example. Issued raised in the context of how the procedure verifies requirements, should be analyzed using the IVV Justification View.

5.4.3 IVV DOCUMENT VIEW

This view is useful to initiate a standard DOORS export to Word (File, Export, Microsoft Office, Word). The Document Style is used to pick up the Paragraph Styles in the template (e.g. normal.dot), the PUID is ignored and the Object Header/Object text field is printed

5.4.4 IVV MATRIX VIEW

The IVV matrix view is useful to show which requirements an IVV procedure is required to verify. It directly provides the identity and text of each requirement to be verified, from these the standard DOORS links can be followed for more information about particular requirements.

5.4.5 IVV PROCEDURES DEFINITION VIEW

This view is used for the initial definition of the IVV procedure, particularly when agreeing the IVV solution with a customer or a supplier. In addition to the procedure PUID and Object Text, it shows the attributes most relevant to this phase of a project. It is probably the view that may need to be exported, printed and made contractual.

5.4.6 IVV PROCEDURES PLANNING VIEW

The IVV Procedures Planning View is useful during the progress of the contract to assist in the management of the IVV planning and preparation phases. In addition to the procedure PUID and Object Text, it shows the attributes most relevant to this phase of a project

5.4.7 IVV JUSTIFICATION VIEW

The Justification view is a useful view to use when analyzing why particular IVV procedures are claimed to verify particular requirements. The descriptions of any issues previously raised are directly presented; it is necessary to follow the DOORS link to determine the status and decision of each issue.

Because the view is derived from traceability via outgoing "is justified by" links, it is most useful when analyzing issues that may have been raised in the context of how the

procedure verifies requirements. Any issues that may have been raised over the allocation of test equipment, for example, should be analyzed using the IVV Associated Issues View.

5.4.8 IVV TEST EQUIPMENT VIEW

The IVV Test equipment view is used to analyse what test or other support equipment and/or facilities have been allocated to a procedure. The allocation may be amended by adding or deleting links, remember to define the default link module first, and to source any new links in the other (PBS) module.

The descriptions of any test equipment or facility are directly presented; it is necessary to follow the DOORS link to determine any more detail about the support item. These descriptions are derived from incoming ”is allocated to” links.

6 Managing your data from a document point of view

6.1 Introduction

In the IRDRMFAO workbench, the data reference is the DOORS database, it is not any documents produced from the database. Data, however, may be imported from word processed documents (e.g. import an external customer document), or data may be output to a word processor and formatted in a pleasing style (e.g. to produce final or intermediate results).

For some people, it could be more convenient to write the major part of a new document outside DOORS with a word processor tool and then import it. Once such a document is imported into DOORS, the DOORS Database becomes the reference. As far as possible, textual modifications should be done within DOORS. Editing DOORS reference data by export from DOORS to a publication tool, and re-importing, is always possible but probably never cost-effective and should be done for consulting only.

Recognising this position, IRDRMFAO provides specific functionality to improve the exchange of DOORS data with external document formats.

6.2 The document view

According to the introduction § 2.3 "DATABASE AND DOCUMENT APPROACHES" each RMF module type provides a view named "Document view" that is set as the default view. This view displays the paragraph style, the PUID and the object text attributes, as shown in Figure 8.1 below These attributes are the there by default in the standard RMF modules and support the standard DOORS export to Word.

The view allows the paragraph style assigned to each object to be viewed, in the column headed "Document Style". Provided this style name exists in the template of the document being exported to, each object will be displayed and printed according to that style definition in Word.

The style for each object may be changed in the Document View using the IRDRMFAO tool "Manage Object", in the "Edit Style" tab.

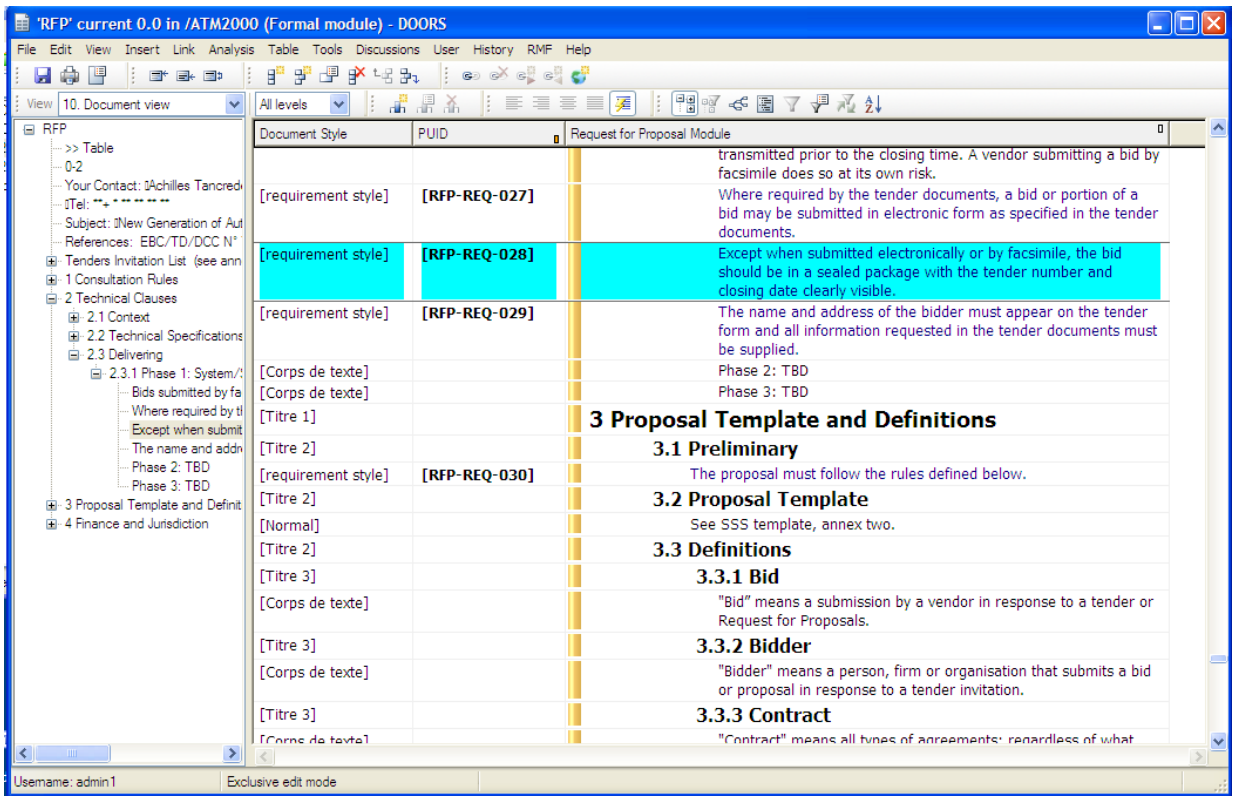


Figure 50 : Example of Document View

6.3 Editing paragraph styles

6.3.1 Using IRDRMFAO tool “Manage Object”, “Edit Style” tab

The paragraph style for an object, can be modified by using the IRDRMFAO utility “Manage Objects” (the “Edit Style” tab), which provides an enumerated drop down list to choose from. It is also possible to change the object attribute containing the text value, that can be the “Object Heading” or the “Object Text” attribute. The chosen style is applied to the chosen attribute.

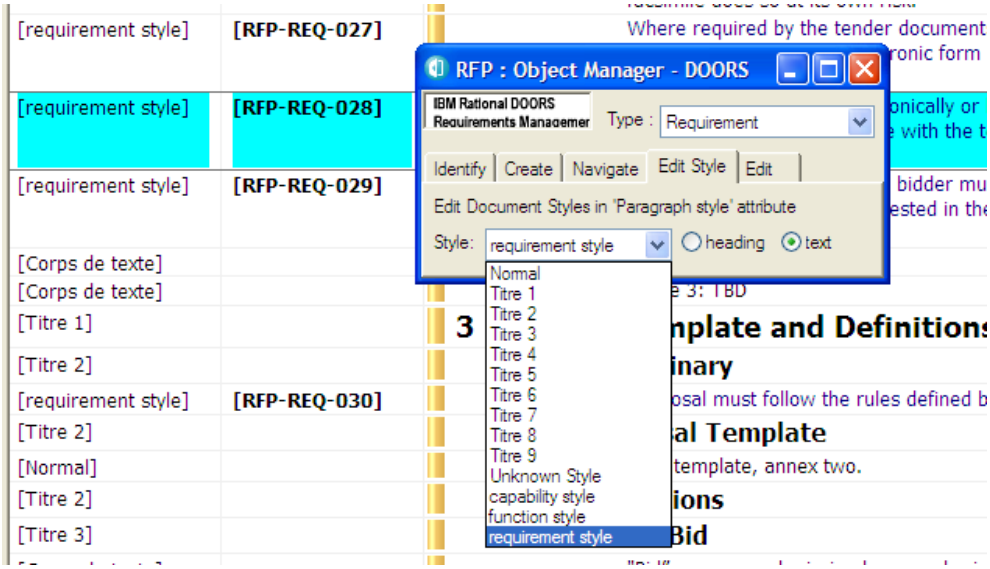


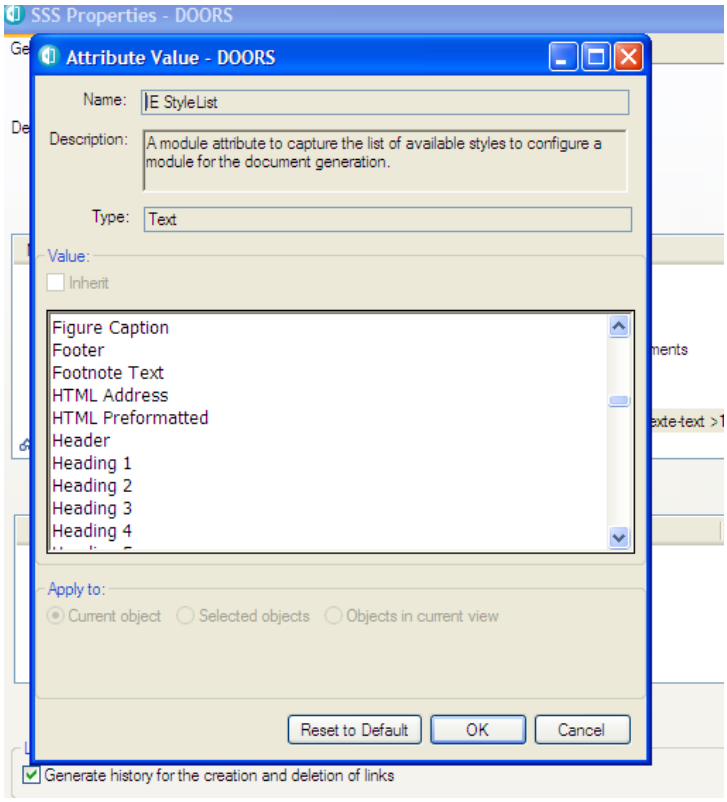
Figure 51 : Manage Object dialog box, Edit Styles tab

The list of paragraph styles offered by the utility comes

- from the Module definition object in the Project Profile module. (The Object definition objects in the Project Profile module define the default paragraph style used when a RMF object is first created or identified). By this means, the enterprise or project documentation formatting style is enforced by IRDRMFAO.

and, also

- from the Module attribute "IE StyleList" value that allows you to let additional styles be available in this module. You may enter those additional styles by using the "Configure Module" tool to initialize the list from an existing Word template:



6.3.2 The DOORS standard tool "Edit paragraph style attribute"

DOORS supports the concept of assigning a different paragraph style to each attribute in each object, but this feature is not encouraged under IRDRMFAO. IRDRMFAO recommends a single style for all attributes within each object type, using only the styles defined in the Project Profile module.

The "Edit Paragraph Style Attribute" DOORS tool allows the user to type in Style names for individual attributes. For more information, refer to the Erreur : source de la référence non trouvée.

6.4 Exporting DOORS data into a WORD document

There are two ways to export data to WORD: (1) using the standard DOORS Word Exporter or (2) using the IRDRMFAO Enhanced Word Exporter. Briefly, use the first one if you want a quick export but with limited possibilities and if you are not too demanding on the final result. Use the second, if you want to build complex documents, which are updated and edited regularly, with a nice final result and needing practically no retouching on the output.

In the two cases, you need to have a Word environment installed in your computer. DOORS needs to execute Word and to communicate with it to be able to generate the document. This principle is also the main limitation of the functionality: the integration between Word and DOORS is based on the Word VBA API, not on the .doc or .rtf file

format. It is not possible to replace Word with any other tool. Also, all the information is sent from DOORS to Word by using the clipboard. The computer is not usable during the execution of the operation. On big documents, the operation may be very long and consume a lot of memory. This can be a reason of failure of the generation, without any real solution, except to split the generation is several pieces.

6.4.1 The standard DOORS WORD exporter

The standard DOORS "Export to Word tool" creates a Microsoft Word document, and exports the current view to it. The structure of the document is the same as the structure of the current view.

You can export in Table format (the document looks like the view you are exporting) or export in Book format (Object Text and Object Headings only, with style applied).

Example:

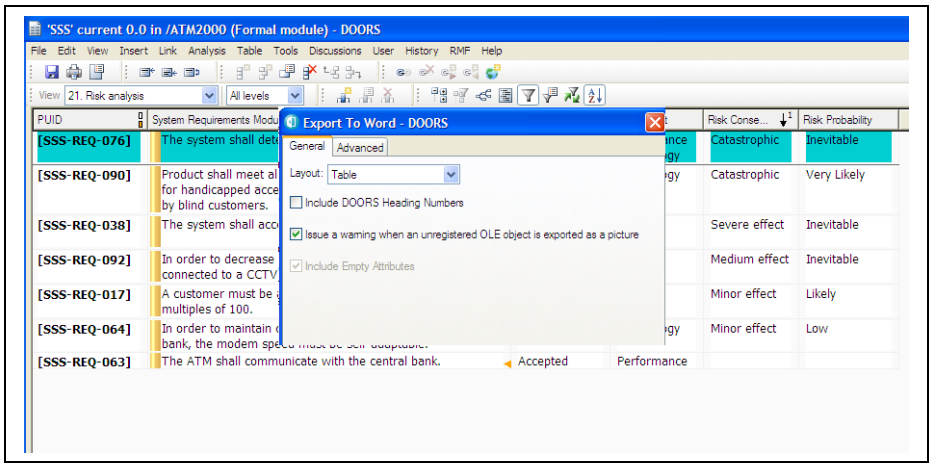
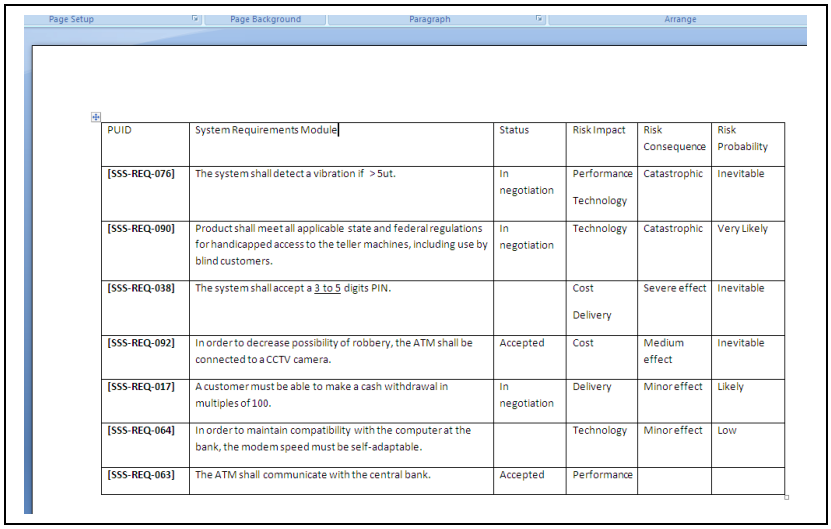


Figure 52 : View to export



PUID	System Requirements Module	Status	Risk Impact	Risk Consequence	Risk Probability
[SSS-REQ-076]	The system shall detect a vibration if > 5ut.	In negotiation	Performance Technology	Catastrophic	Inevitable
[SSS-REQ-090]	Product shall meet all applicable state and federal regulations for handicapped access to the teller machines, including use by blind customers.	In negotiation	Technology	Catastrophic	Very Likely
[SSS-REQ-038]	The system shall accept a 3 to 5 digits PIN.		Cost Delivery	Severe effect	Inevitable
[SSS-REQ-092]	In order to decrease possibility of robbery, the ATM shall be connected to a CCTV camera.	Accepted	Cost	Medium effect	Inevitable
[SSS-REQ-017]	A customer must be able to make a cash withdrawal in multiples of 100.	In negotiation	Delivery	Minor effect	Likely
[SSS-REQ-064]	In order to maintain compatibility with the computer at the bank, the modem speed must be self-adaptable.		Technology	Minor effect	Low
[SSS-REQ-063]	The ATM shall communicate with the central bank.	Accepted	Performance		

Figure 53 : Generated Word document

The generic document mimics the DOORS view content and layout.

6.4.2 The Enhanced Word Exporter (WEXP)

The Enhanced Word Exporter (WEXP) is a powerful document generation tool. It needs to be configured for the particular document type you want to create, but once this has been done, the exporter is simple to invoke. This makes it ideal for reports that need to be re-issued at regular intervals.

There are two aspects to the configuration of the customised exporter:

- setting up an appropriate Word template (.dot) file,
- setting up appropriate views and attributes (at both module and object levels).

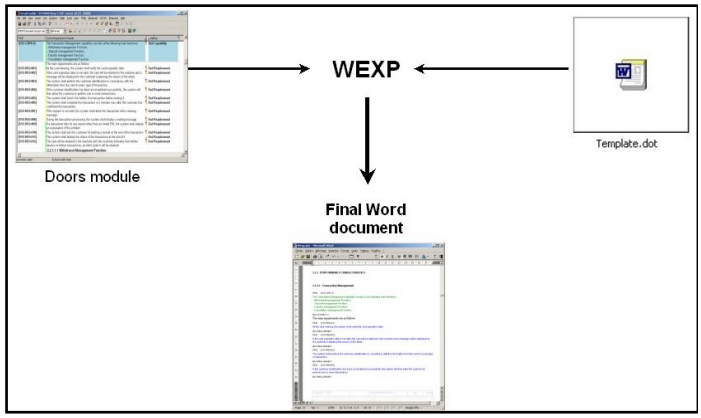


Figure 54 : WEXP representation

To get detailed information about how configuring a template and a module for WEXP usage, you should consult the WEXP manual.

Example:

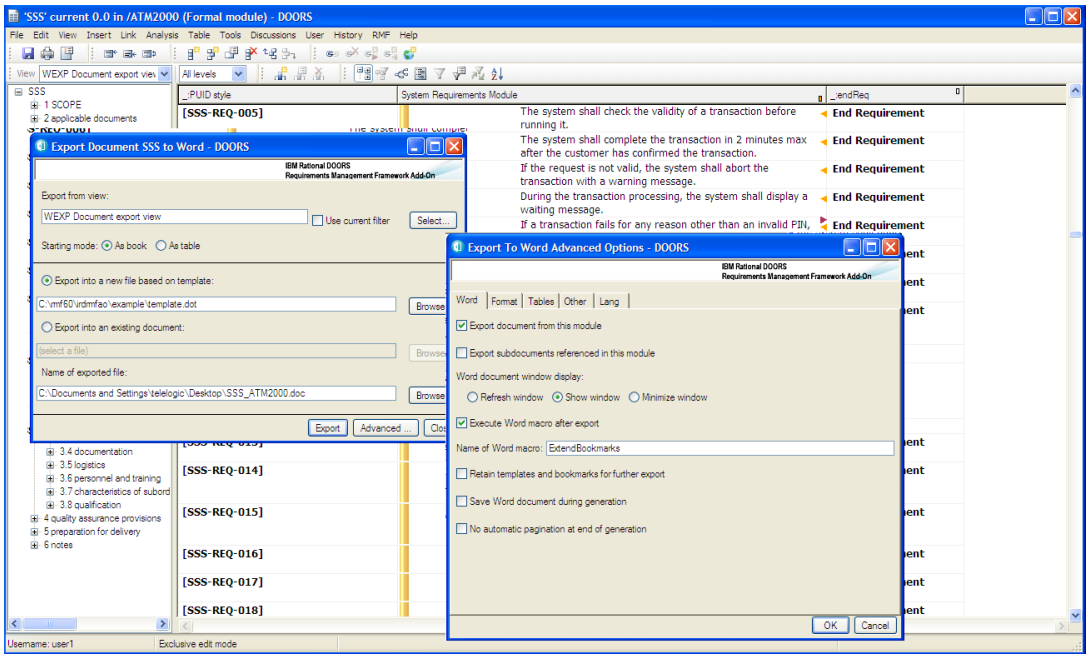


Figure 55 : View to export (configured for WEXP)

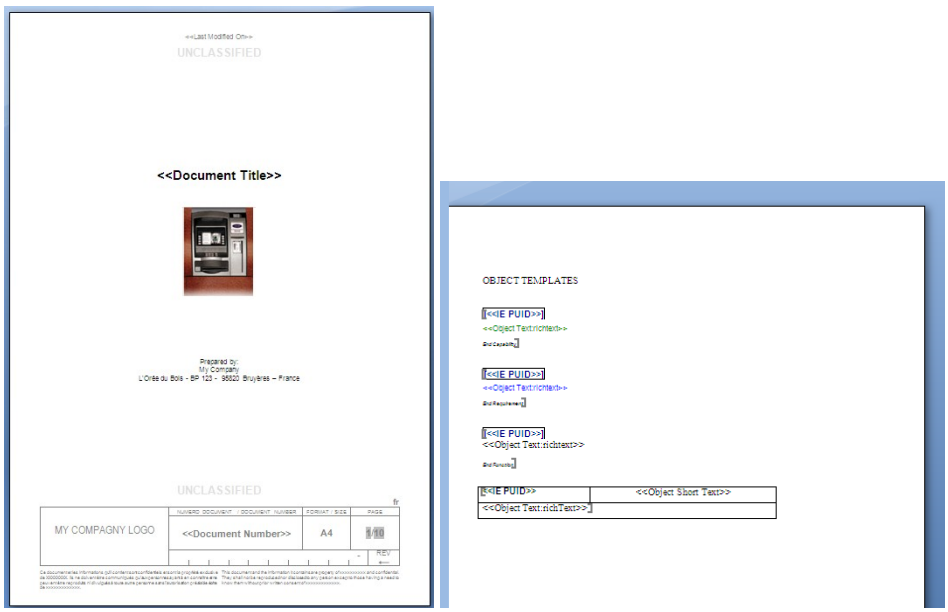
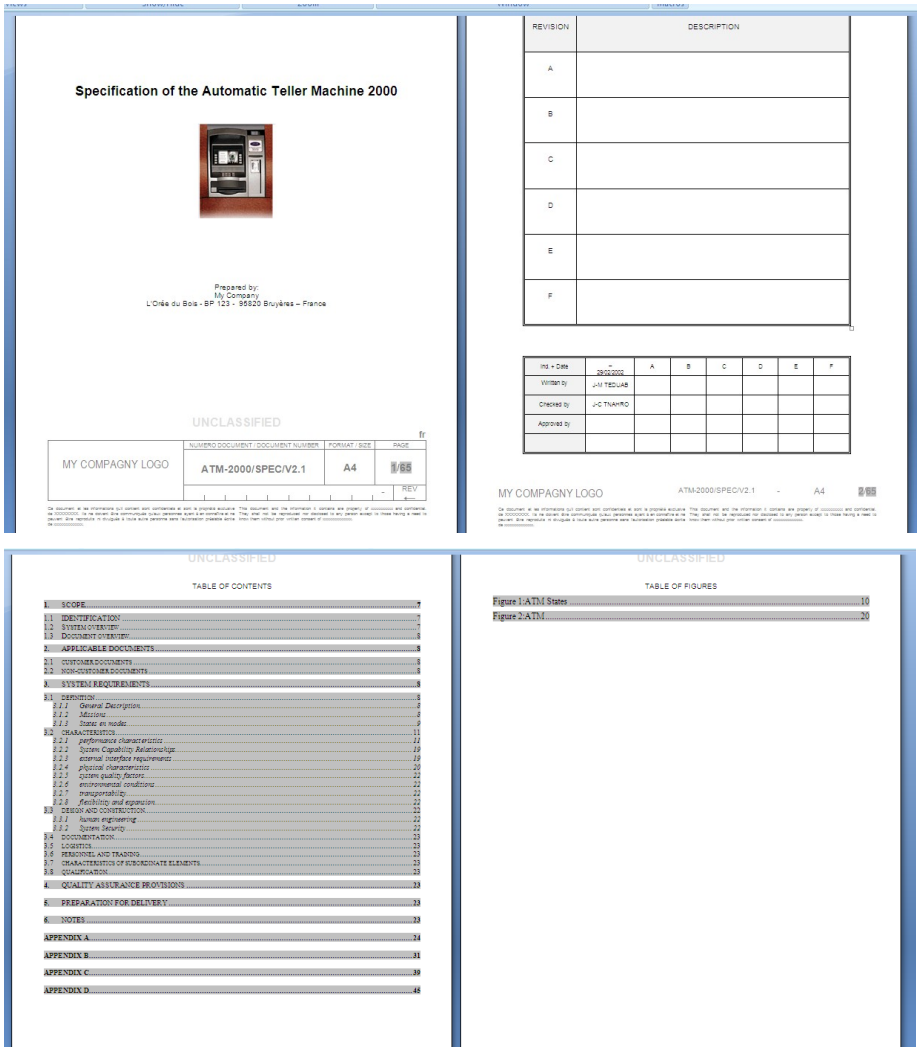
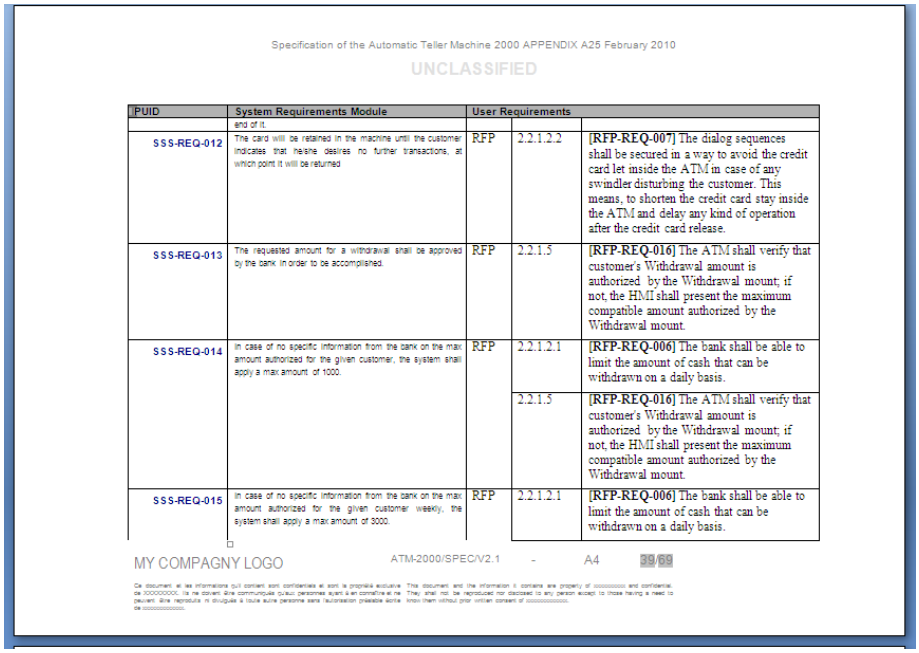


Figure 56 : Template Word (configured for WEXP)



...



Specification of the Automatic Teller Machine 2000 APPENDIX A25 February 2010

UNCLASSIFIED

PUID	System Requirements Module	User Requirements		
	end of it.			
SSS-REQ-012	The card will be retained in the machine until the customer indicates that he/she desires no further transactions, at which point it will be returned	RFP	2.2.1.2.2	**[RFP-REQ-007]** The dialog sequences shall be secured in a way to avoid the credit card let inside the ATM in case of any swindler disturbing the customer. This means, to shorten the credit card stay inside the ATM and delay any kind of operation after the credit card release.
SSS-REQ-013	The requested amount for a withdrawal shall be approved by the bank in order to be accomplished.	RFP	2.2.1.5	**[RFP-REQ-016]** The ATM shall verify that customer's Withdrawal amount is authorized by the Withdrawal mount; if not, the HMI shall present the maximum compatible amount authorized by the Withdrawal mount.
SSS-REQ-014	In case of no specific information from the bank on the max amount authorized for the given customer, the system shall apply a max amount of 1000.	RFP	2.2.1.2.1	**[RFP-REQ-006]** The bank shall be able to limit the amount of cash that can be withdrawn on a daily basis.
			2.2.1.5	**[RFP-REQ-016]** The ATM shall verify that customer's Withdrawal amount is authorized by the Withdrawal mount; if not, the HMI shall present the maximum compatible amount authorized by the Withdrawal mount.
SSS-REQ-015	In case of no specific information from the bank on the max amount authorized for the given customer weekly, the system shall apply a max amount of 3000.	RFP	2.2.1.2.1	**[RFP-REQ-006]** The bank shall be able to limit the amount of cash that can be withdrawn on a daily basis.

MY COMPAGNY LOGO ATM-2000/SPEC/V2.1 - A4 39/69

Ce document, et les informations qu'il contient sont confidentiels et sont la propriété exclusive de XXXXXXXXX. Ils ne doivent être communiqués qu'aux personnes ayant à en connaître et ne peuvent être reproduits ni divulgués à toute autre personne sans l'autorisation préalable écrite. This document and the information it contains are property of xxxxxxxxxx and confidential. They shall not be reproduced nor disclosed to any person except to those having a need to know them without prior written consent of xxxxxxxxxx.

igure 57 : WEXP generated document

7 Data model customization

7.1 DESCRIPTION OF A RMF PROJECT STRUCTURE

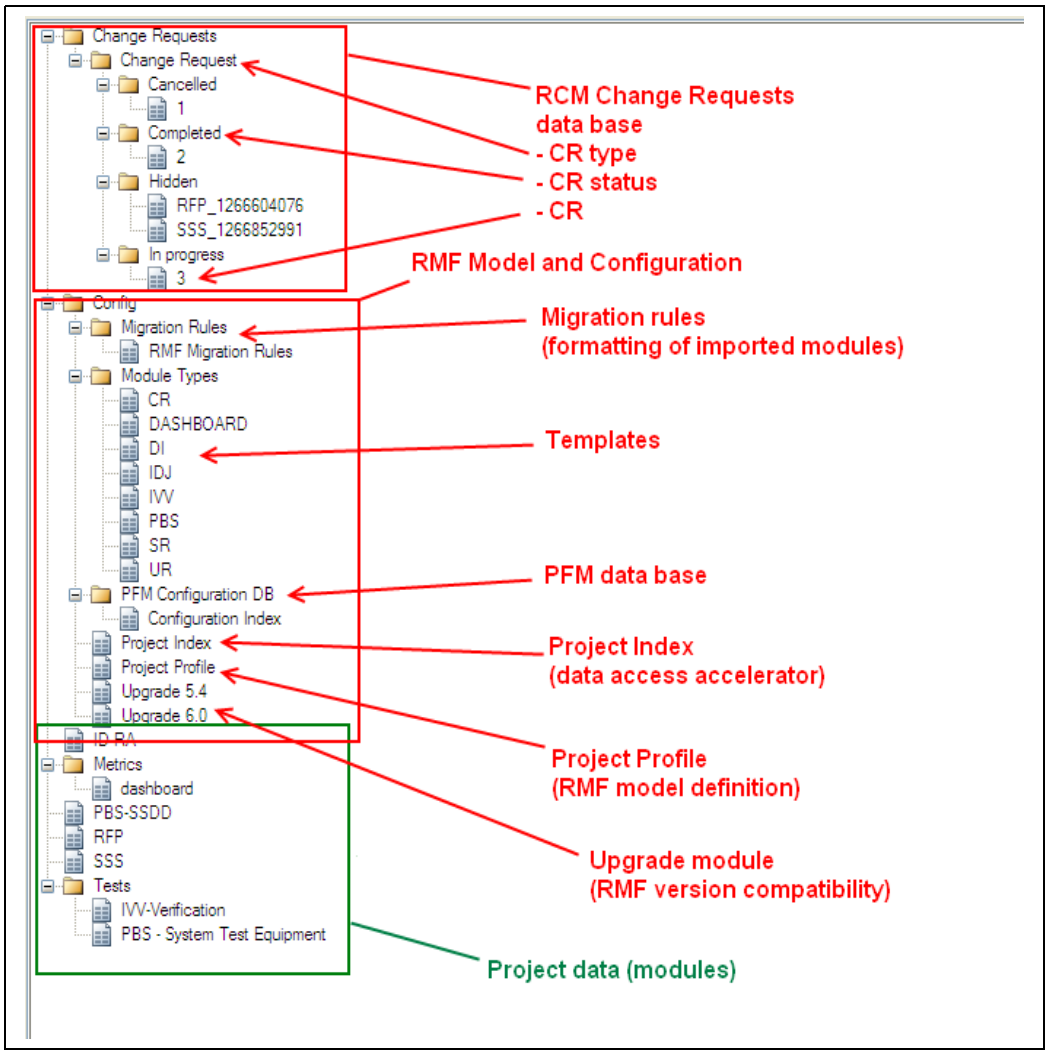


Figure 58 : Example of RMF project structure

When you create a new RMF project, you have a "Config" folder under the root project folder. This folder contains the description of the RMF model. In the "Config" folder, some data will be dynamically created or modified, some other are static. But in any case, all the modifications should be executed by using the RMF tools, or by somebody that has a very good knowledge of RMF.

Some other items may also be created by RMF in the "data" part of the project:

- Predefined link modules. In general all the link modules are predefined and located at the root folder of the project.
- Predefined folders and modules. For example, the "Metrics" folder with some predefined "Dashboard" modules, may be used to deploy dashboards with predefined metrics associated with your model.
- Predefined data folders and modules. For example, you may decide that the product breakdown structure of tour product may be used to organize your

project, and each element of the breakdown structure is used to create a folder to contain the modules associated with the product element.

The "Change Requests" is also an example of predefined folder. It is used by the RCM component, only if you configure the definition of this folder for your project and you activate RCM. This folder will require specific access rights, different from the "standard" access rights, and also different from the "Config" folder access rights. The name and the localization of the folder may be change, but the content of the folder is managed by RCM and must never be modified manually.

The "Config" folder contains different elements:

- **Project Profile** module. This module is the description of your RMF model. It contains the declaration of all templates, module types, objects, relationships and some other model definitions. The project options defined at project configuration are saved as module attributes of the Project Profile.
- **Module Types** folder. This folder contains the list of templates. The list of templates and the template content should be consistent with the content of the "Project Profile". The RMF Explorer is able to check this consistency. The definitions (types, attributes, views) associated with a given module type are defined into the associated template.
- **Migration Rules** folder. This folder contains the modules describing the different migration rules.
- **PFM Configuration DB** folder. This folder contains a formal module and a link module used to save the PFM information. It is used only if PFM is activated in your project. All the information associated with one family is saved into one PFM DB.
- **Project Index** module. This module is created and initialized with the "Manage Index" tool. It can be deleted and re-initialized at any time. It collect information from the project link modules and formal modules, allowing to get the information faster than having to open each formal or link module to get it.
- **Upgrade *.*** module. These modules are used to indicate the compatibility of the Project Profile module with a given version of RMF. For example, if you create a project with RMF 5.4, before using RMF 6.0 on it you have to execute the "Upgrade" operation. At the end of the upgrade, the module "Upgrade 6.0" will be created, to indicate that no more upgrade is required (the upgrade is always at model level only, it is your responsibility to upgrade or not the project data).

If you want to modify your data model, you will have to edit manually information in the "Module Types" folder (i.e. the template modules) and in the "Project Profile" module.

7.2 ADAPTING MODULE ATTRIBUTES

The default attributes proposed in module types may be not relevant for your project. Then, according to your need, you can delete, modify or add attributes in RMF module types, except for reserved RMF attributes as described below.

Reserved RMF attributes

All the formal modules of an RMF project derived from a RMF template, will have at least four main, reserved, attributes.

The IRDRMFAO tools use these four main attributes, and they should not be modified manually by standard users (you may eventually redefine the name of these attributes, but you should indicate the new names in the file:

$IRDRMFAO/lib/dxl/addins/irdrmfao/startup/userdefs.inc

Module attributes:

Each module has the following attribute defined as a DOORS module attribute:

Attribute name	Type	Description
IE Mod Type	enum	This attribute represents the type of the current module. The *Project Profile* module contains the list of all the possible values defined for the current project. But for a given module, this attribute will have a value assigned once and for all. The value should be one from the list defined in the *Project Profile*.

Object attributes:

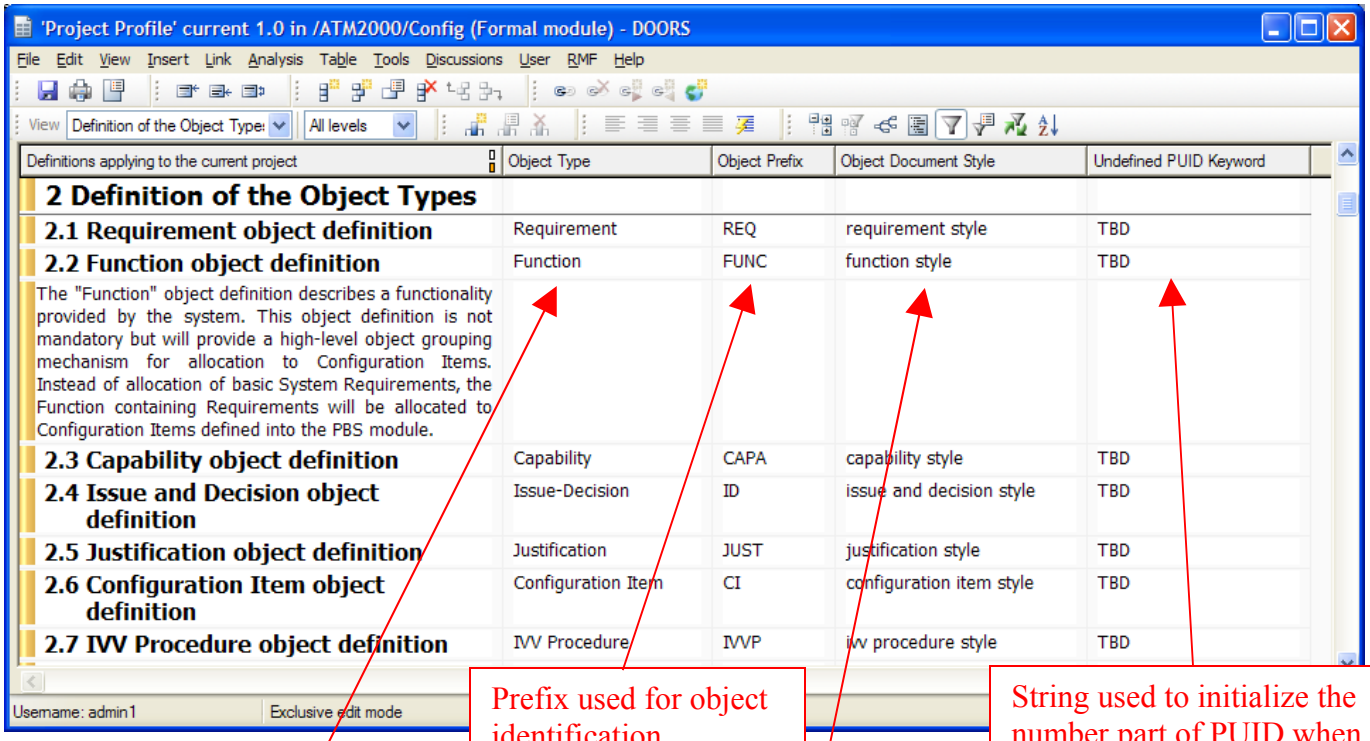
Each module has the following attributes defined as DOORS object attributes

Attribute name	Type	Description
IE Object Type (**inherited** attribute)	enum	This attribute represents the type of the object. The *Project Profile* module contains the list of all the possible values defined for all the modules within the current project. The value should be one from the list defined in the *Project Profile*.
IE PUID **(never inherited)**	string	Project Unique ID. This attribute represents the identification of the object. It is managed either: - automatically using IRDRMFAO utility *Renumber All Objects*, - or by hand.
Paragraph Style	string	This attribute is not an RMF attribute. It comes from DOORS. You can capture the style formatting of paragraphs in your Word document, and store it in the *Paragraph Style* attribute. If you later export the module to a new Word document, the styles named in the *Paragraph Style* attribute are used to format the document.

7.3 ADAPTING PROJECT PROFILE

The *Project Profile* formal module describes the available templates, object types and module types in the project. The Project Profile module must be modified, as described below, if new object, module type or template is added to the project.

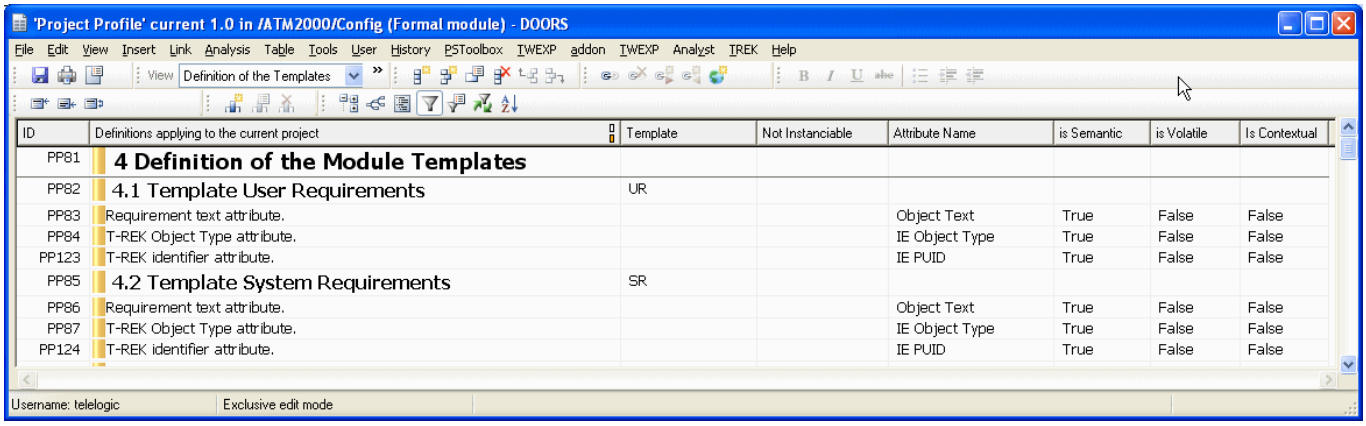
- The "object types" represent all the data suitable for System Engineering, like requirements, capabilities, issues and decisions, etc. The possible values for Object Type are: Requirement, Function, Capability, Issue-Decision, Configuration Item, IVV Procedure.

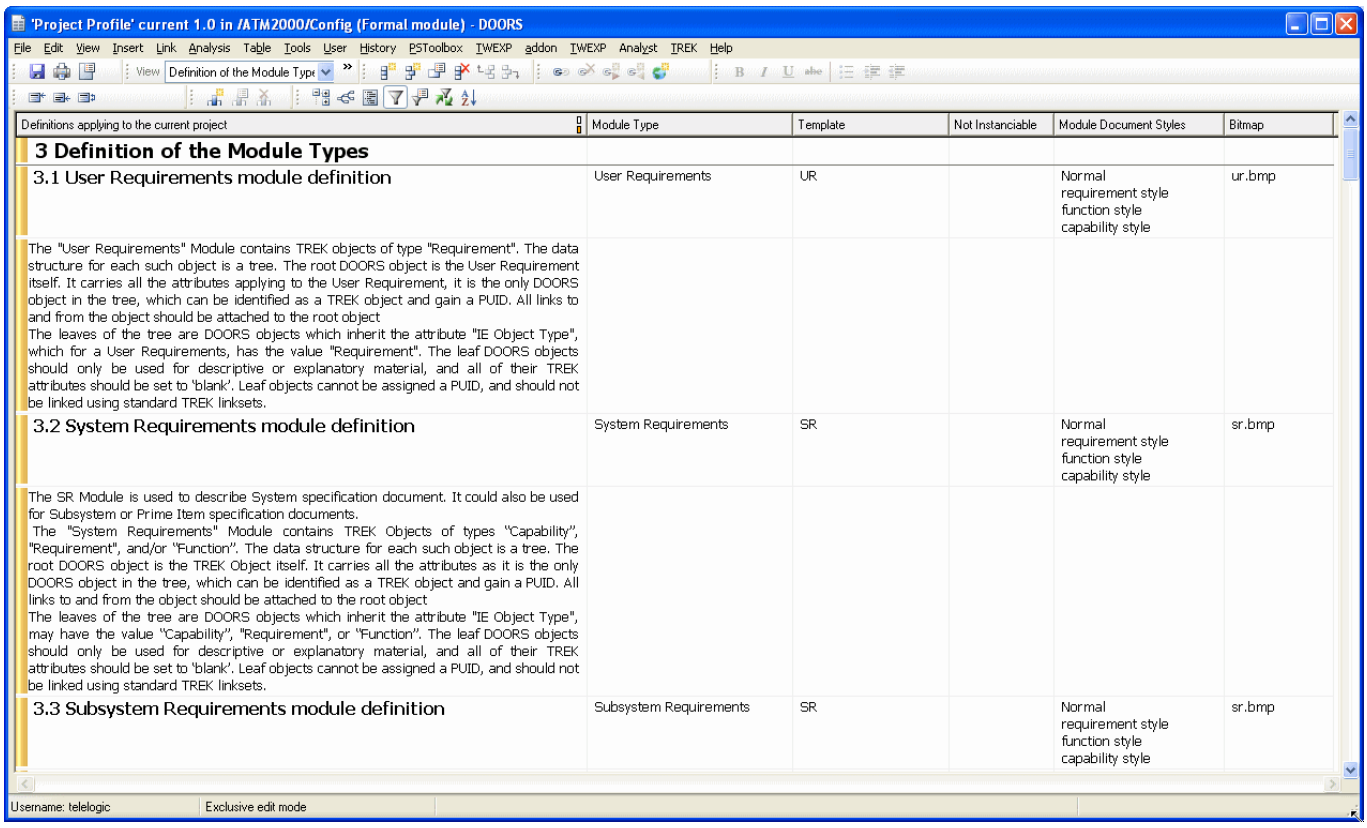


- The "templates" are the available models when creating a new module in the project.
- Each template has one or several "module types" corresponding to ageneric use of the template.

  Exemple:

  - The Module Types of the template "SR" are:
    - System Requirements,
    - Subsystem Requirements,
    - Prime Item Requirements,
  - The Module Types of the template "IDJ" are:
    - Issues and Decisions,
    - Operational Concepts.



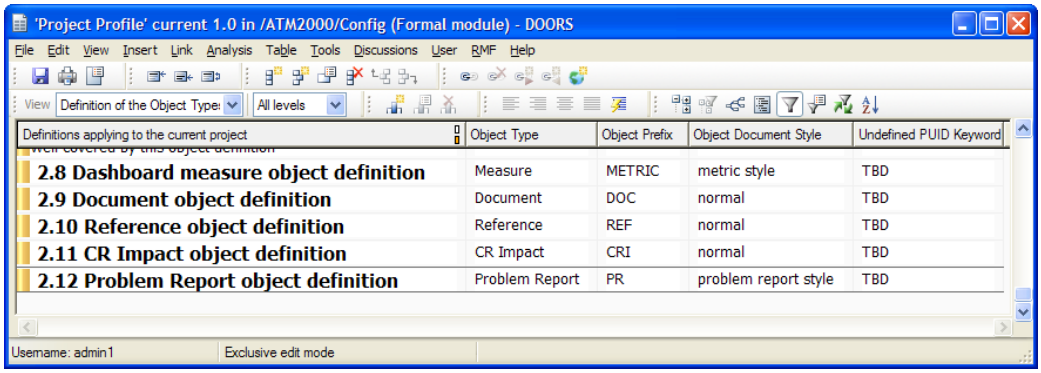


7.3.1 ADD A NEW OBJECT TYPE

To add a new object type, follow the steps below. The example shows the creation of an object type "Problem Report".

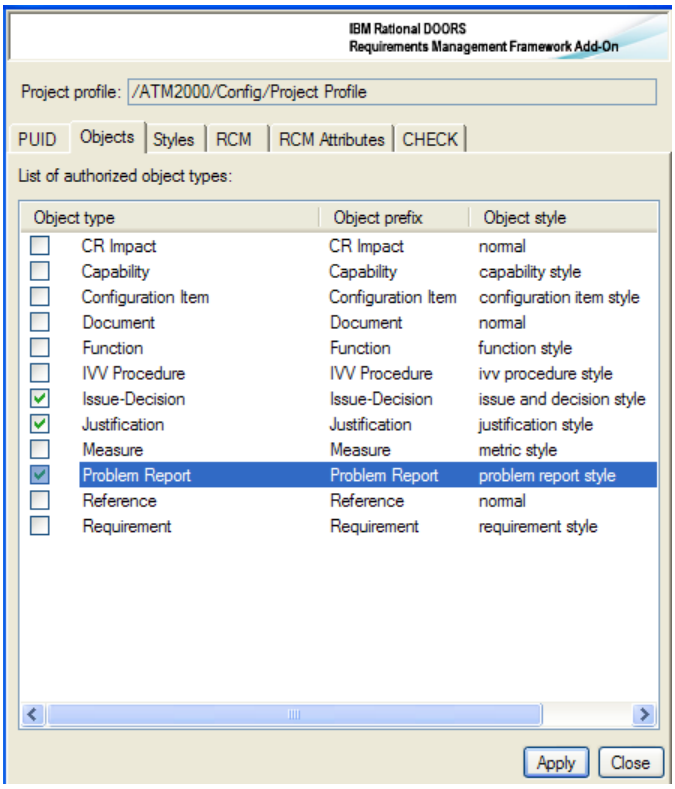
- Within DOORS, open the *Project Profile* module and select the view "definition of the object types".
- Select the last object, and create a new object (Menu "Object->New"). Fill the attributes:

  Text = "Problem Report object definition",
  Object type = "Problem Report",
  Object prefix = "PR",
  Object document style = "problem report style",
  Undefined PUID keyword = "TBD"



- Save the *Project Profile* module.
- Now, to use this new object, you have to declare it in each module you want to use it. Let's take as example that you want to use it in an existing module derived from the UR template.

- Open the UR type module, and run the menu “RMF ->Configure module”.
- Click on the type name you want to use in this RMF module



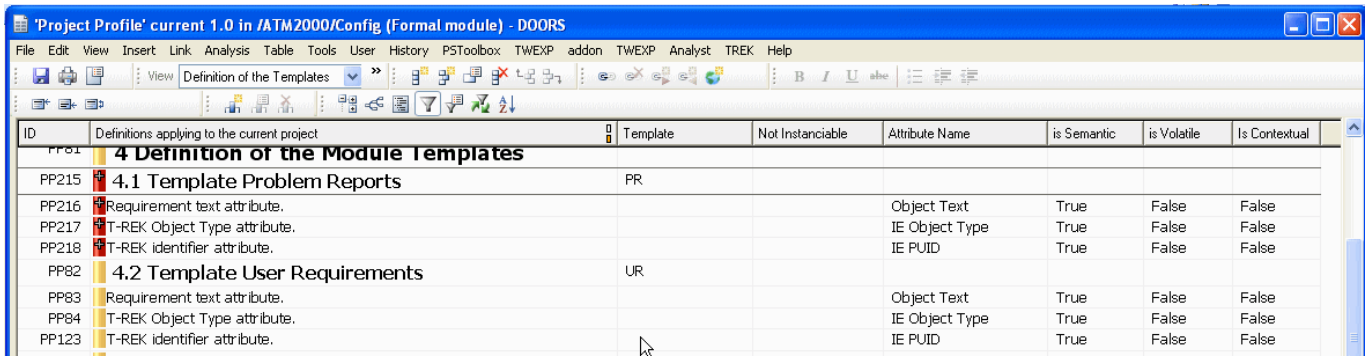
You may also want to add this capability to all modules derived from the UR template. In this case, use the module configuration operation on the UR template module, and then synchronize all your UR modules with the “Create/Tag a RMF module” operation.

7.3.2 ADD A NEW TEMPLATE

IRDRMFAO provides several standard templates: UR, SR, PBS,…but if you need to add a specific template for your project, or if you want for example have a different model for a “System Requirements” and a “Sub-system Requirements” module, follow the steps shown below. The example shows the creation of a new template named “PR” for "Problem Report".

- Within DOORS, open the *Project Profile* module and select the view "Definition of the templates".
- Duplicate the object 4.1 with its descendatnts (Copy with hierarchy + Paste)
- Modify the new first pasted object as below :

  Heading = "Template Problem Reports",
  Template = "PR"
  Not instanciable = <leave blank>



- Optionally add new semantic attributes. Do not remove one of the tree default semantic attributes (Object Text, IE Object Type, IE PUID)

  Object Text = "attribute description",
  IE Attribute Name = <attribute name>,
  IE Is Semantic = True

- Save the *Project Profile* module.

The semantic attribute declaration is used by RCM, a semantic attribute is under RCM control, and can't be modified outside of a Change Request context, if the module is under RCM control.

A template must have at least one module type. So you must also declare a module type. In the example below, the module type "Problem Reports" will be created for the template "PR".

When you declare a template into the Project Profile, you have also to define it into the "Module Types" folder:

- Create a new module of name "PR" in the "Module Types" folder
- Edit this new module and add the mandatory attributes. Define the modules types with the "Configure Module" operation.
- Define the other attributes
- Define the views

7.3.3 ADD A NEW MODULE TYPE

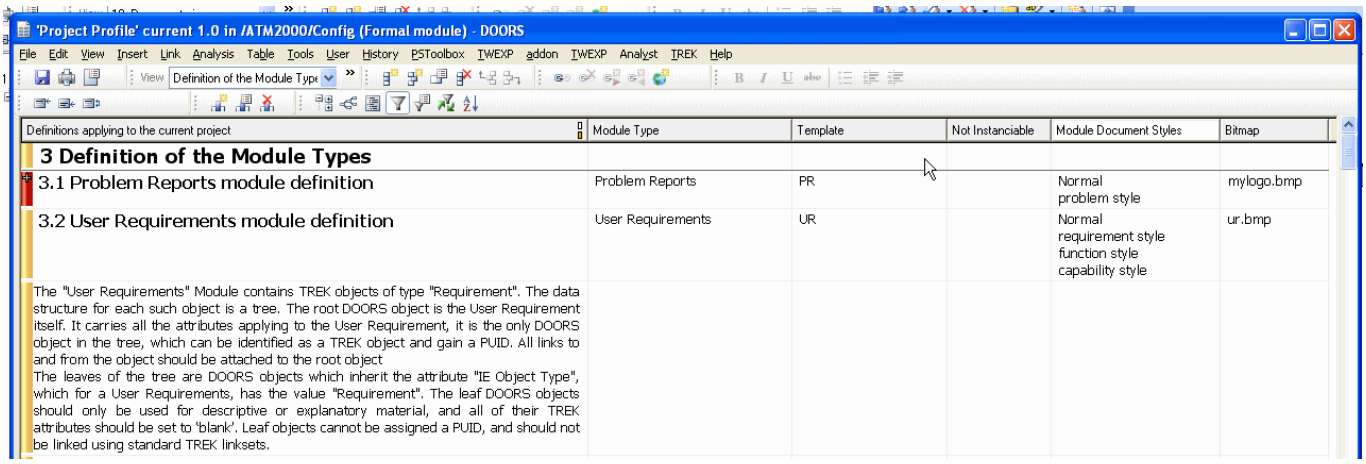
The example shows the declaration of a module type named "Problem Report" available in the template "PR".

- Within DOORS, open the *Project Profile* module and select the view "Definition of the module types".
- Select the last object, and create a new object (Menu "Object->New"). Fill the attributes:

  Object Heading = "Problem Report module" definition",
  IE Module type = "Problem Reports",
  IE Template = "PR",

  IE Module word styles = <list of the styles that can be chosen>,

  IE Bitmap = <bitmap file name (for the explorer)>



- Save the *Project Profile* module.

From now, you are able to create a formal module of type "Problem Reports" using the template “PR”.

You can create the template “PR” from an existing RMF standard module or create it from scratch but do not forget to create and set mandatory RMF attributes. Refer to *§ Adapting module attributes*. In particular, the attribute **"IE Mod Type"** defined at the module level with the value "Problem Reports", and the attribute **"IE Object Type"** defined at the object level with at least the value "Problem Report" in our example.

You may also set a value to the attribute “**IE Bitmap**” to define the bitmap associated with the module type and displayed by the **Explorer**.

7.3.4 ADD A NEW RELATIONSHIP

RMF’s tool named “Explorer” enables to check that the relationships in your project conform to a definition made into the Project Profile. It also provides several means to fix any problem.

You may define a new relationship by editing the Project Profile module:

- Add a new object, in the chapter “Definition of the Relationships”
  - Set the attribute “IE Relationship” with the name of the relationship (this attribute is mandatory).
  - Set the attribute “IE Link is semantic” to “true” if those links are semantic (as opposed to links created for technical reasons, such a comparison links).
  - Set the attribute “IE Link can be suspect” to “true” if you want that RCM let those links become suspect when the object at the other end has been modified (a new version has been created).
  - Optionaly set the text and heading of the object (to describe the purpose of the relationship)
- If the relationship is semantic, then, for each linkset (i.e. a pair source module type, target module type), create a new object under the relationship object. Enter in the heading a short description of the linkset. Set the attribute “IE Relationship” with the name of the relationship. Set the attribute “IE Source Module” with the name of the source module type. Set the attribute “IE Target Module” with the name of the target module type.
- You may add text objects under the relationship object and the linkset objects to enter comments.

View of the project profile:

Vue Definition of the Relationship: Tous les niv

Definitions applying to the current project	Source Module	Relationship	Target Module	Mapping	semantic	suspect
5 Definition of the Relationships				Many-to-Many		
5.1 Refines relationship definition		refines		Many-to-Many	False	True
5.2 Satisfies relationship definition		satisfies		Many-to-Many	True	True
The **satisfies** relationship is used to capture the dependencies between requirements, from one level to another level (for example from the system requirements to the user requirements).				Many-to-Many	True	True
5.2.1 System to User	System Requirements	satisfies	User Requirements	Many-to-Many	True	True
5.2.2 Sub System to System	Subsystem Requirements	satisfies	System Requirements	Many-to-Many	True	True
5.2.3 Prime item to Sub System	Prime Item Requirements	satisfies	Subsystem Requirements	Many-to-Many	True	True
5.3 Verifies relationship definition		verifies		Many-to-Many	True	True
The **verifies** relationship is used to capture the dependencies between the **verification procedures** and the **requirements.**				Many-to-Many	True	True
5.3.1 Verification to SR	Verification Procedures	verifies	System Requirements	Many-to-Many	True	True
5.3.2 Integration to SR	Integration Procedures	verifies	System Requirements	Many-to-Many	True	True
5.3.3 Validation to UR	Validation Procedures	verifies	User Requirements	Many-to-Many	True	True
5.3.4 Installation to UR	Installation Procedures	verifies	User Requirements	Many-to-Many	True	True
5.4 Is allocated by relationship definition		is allocated by		Many-to-Many	True	True

- Save the *Project Profile* module.

The source and the target of a linkset may be a Module Type (e.g. "System Requirements") or a Template (e.g. "SR"), depending on the genericity level of the association.

7.3.5 CHECK THE MODEL

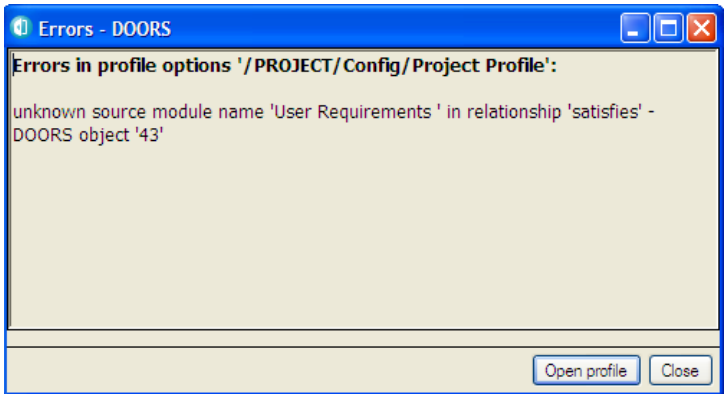
The edition of the Project Profile is completely manual, and it is very easy to make mistakes. For example, if you write "System Requirements " and not "System Requirements", your model is incorrect.

To check the consistency of the model use the Explorer. During the initialization of the tool, the consistency of the references is checked, and in case of error, a message is displayed.

Example: the target module "User Requirements" is written with a wrong space character

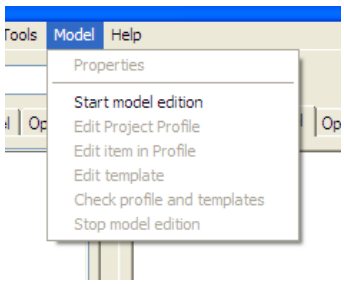
Definitions applying to the current project	Source Module	Relationship	Target Module	Map
5.1.1 System to User	System Requirements	satisfies	User Requirements	Ma
5.1.2 Sub System to System	Subsystem Requirements	satisfies	System Requirements	Ma
5.1.3 Prime item to Sub System	Prime Item Requirements	satisfies	Subsystem Requirements	Ma

When executing the Explorer, this message is displayed:

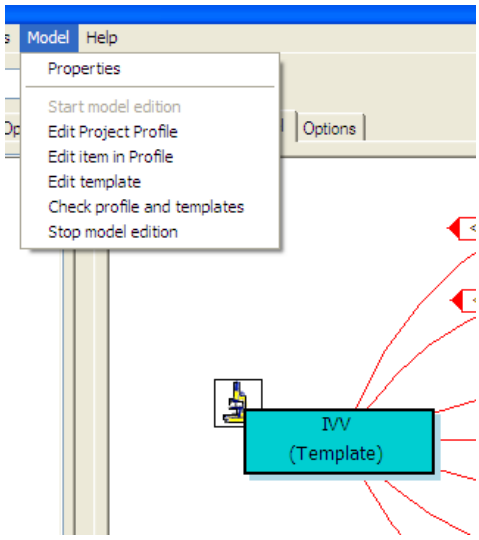


You have also the possibility to edit the Project Profile from the Explorer:

- Execute the operation "RMF -> Explore -> Model", or switch to the "Model" tab in the Explorer
- Execute the operation "Start model edition" from the Model menu of the Explorer



- Confirm the backup proposal. This feature will allow you to return to the original model version in case of error.
- Execute one of the Edit operations. You may edit the Project Profile module or one of the template;



- At any time you can execute "Check profile and templates" to validate your modification.
- When the edit session is terminated, execute "Stop model edition". This operation will check again the model, and if all is OK will stop the edition. You may delete or not the backup (i.e. the previous version of the model).

In any case, all your modifications must be ascendant compatible, because of the existing data. It is your responsibility to migrate your data in case of not compatible modification.

For example, changing the type of an attribute in a template (from Text to Enumeration) will not be automatically propagated on the data?

7.3.6 ERROR HANDLING

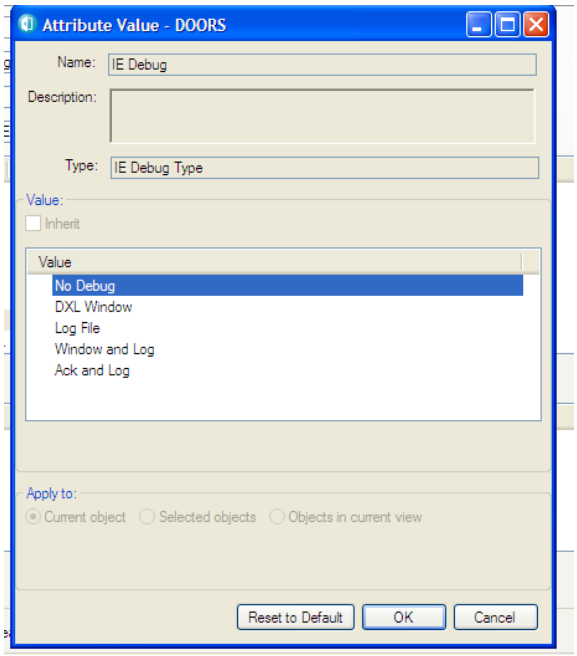
The *Project Profile* formal module also allows switching the IRDRMFAO debug mode on to control the handling of error messages from the utilities.

Use this advanced feature only if you are an experienced user or if your support team wants to track a bug.

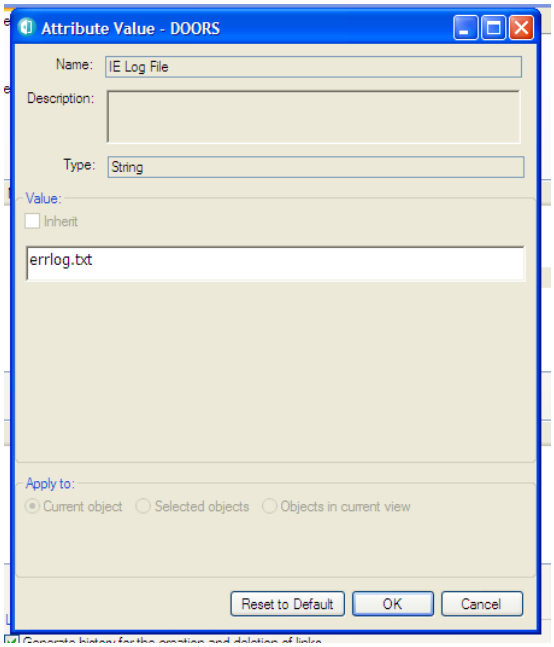
To change the debug mode, edit the *Project Profile* attributes (menu *File->Module properties*…) and set "IE Debug" to

- No Debug (default value),
- DXL window (to make appear the DXL program when a problem occurs),
- Log file (to send the output error messages in a log file),

- Window and Log (to send the output error messages in a log file and on the screen),
- Ack and Log (to send the output error messages in a log file and a screen problem acknowledgement).



Set the attribute "IE Log File" to define the name and the location of the output errors messages (default is "errlog.txt")



8 Import and identification of RMF objects

8.1 Introduction

RMF objects are DOORS objects with specific values in their type and identification attributes. The permissible values for the RMF object type(s) in each RMF module type, are defined in the Project Profile module, see paragraph 4.2.3.

RMF objects are acquired by either 'Identifying' an existing DOORS object, or by 'Creating' a new RMF object, or by a combination of both.

Establishing an appropriate hierarchical structure of the RMF objects is an important issue, and the significant differences between requirement modules and breakdown structure modules are outlined below.

The detailed operation of the IRDRMFAO utilities is described in the Reference Manual, a summary of their usage is described here.

8.2 RMF Object structure within a module

8.2.1 Requirement Type Structures

It is an important **methodology issue that requirement statements do not have a hierarchy**, they must be self-contained verifiable statements. (Any decomposition of the requirement into sub-requirements in lower level entities should be held in different modules, and linked back to the parent requirement object.)

However, it is also necessary to structure long lists of requirement statements into sections and sub-sections for clarity and ease of use.

These two objectives are achieved within IRDRMFAO by using the DOORS concepts of "Heading" or "Normal Text". All requirement statements are "Normal Text", grouped at the same DOORS level under a particular heading.

A DOORS hierarchy of "Headings" (i.e. Object Text is blank) should be established to aid clarity,

IRDRMFAO enforces this methodology in requirement type modules by only allowing "Normal Text" objects (i.e. not "Headings") to be created or identified as RMF Objects, and by not allowing RMF objects to be created or identified if a parent or ancestor object is already a RMF object.

The IRDRMFAO "Manage Objects" tool will create a new RMF object at the same level as the current object if the current object is "Normal Text"; and will create a new RMF object one level below if the current object is a "Heading".

In addition, requirements may be structured into Functions and Capabilities, as described in paragraph 5.4.4.

8.2.2 Breakdown Type Structures

In equipment or system breakdown structure type modules there is no methodology restriction on the object hierarchy, in fact configuration objects may need to be related in more than one hierarchy. For these modules, such as the PBS module type, IRDRMFAO allows the natural DOORS level hierarchy to be used. Additional hierarchical relationships

between the objects can be set up using the "is composed of" or additional project-specific linksets.

Configuration objects may be entered as "Headings", i.e. with the Object Text blank and with the name in the Object Heading attribute. DOORS will then assign outline paragraph numbering and display the hierarchy in the DOORS Explorer window. Note that this numbering may change if items are added, moved or deleted.

If the current object is a "Heading", the IRDRMFAO "Manage Objects" tool will create a new "Normal Text" RMF object at one level below the current object, the new object should then be converted to a heading.

Because of the volatility of the outline numbering, the RMF PUID should be used as the principal referencing system, but note the behaviour of the PUID numbering in a shared-edit environment as described below. Projects should also add an attribute to hold a part number as assigned by the company PDM system.

The DOORS outline hierarchy is recommended for the design hierarchy of a system, and physical structures should be created using explicit RMF "is composed of" links between the objects. Within the design hierarchy, a configuration item should only appear once, within a physical hierarchy it may appear in several places. The use of links internal to the module allows this. If required, a project could define more link types and use them to represent more hierarchies, such as integration threads for example.

8.2.3 Issue/Decision Type Structures

In the common case, Issue/decision type modules are linear lists of objects that do not need an implicit structure. They are accessed via explicit links from relevant requirement or configuration objects in other modules; reports are usually generated using sorting and/or filtering on specific attributes. A long list of objects at a single level is not a problem. One only reason for imposing a structure might be to divide the module into sections to enable multi-user access under DOORS, but this is an implementation solution rather than a methodology imperative. Issue/decision RMF objects should therefore be created as 'Normal Text', with 'Headings' only used if required to divide the module into shareable sections.

In other cases, Issue/decision type modules could be structured to produce document such as Justification documents.

8.3 HOW TO Import a document into DOORS

Often the first action you will carry out will be to import a document into DOORS, this will create a set of general DOORS objects, which must subsequently be "Identified" as RMF objects.

The imported documents can be:

- the customer requirements (into a UR Module) ,
- existing documents in a reuse case (e.g.: the existing specification of a previous system that is close to your customer needs (into an SR Module)).

In all case, you need the electronic format of the document to be available. Only Word documents may be imported, for example PDF documents can't be imported.

The procedure is:

- First, always start by creating an appropriate RMF module type to receive the document,
    - Second, use the standard DOORS utilities to Import (From DOORS menu "File-> Import" or better from WORD application using the "Export to DOORS" icon).

- Or use the RMF utility from WORD, that contains more functionalities

➢ Once the document is imported into a RMF module, it must be structured as described in the previous section and the objects identified as RMF objects.

8.4 HOW TO IDENTIFY RMF OBJECTS: REQUIREMENTS, FUNCTIONS,

8.4.1 Introduction

There are two basic methods of achieving an identified RMF object, as follows:

➢ Identify a "Normal Text" DOORS object, or

➢ Create a new RMF object.

8.4.2 Identifying an existing DOORS object

To identify an RMF object, select a DOORS object or several and activate the utility "Manage Objects" in the current module menu "RMF". Notice that the dialog window stays displayed in front of the screen, allowing quick repeated object identification, scanning the DOORS module. The same dialog box can also be used to identify objects in other modules, and be used to create RMF objects as well.

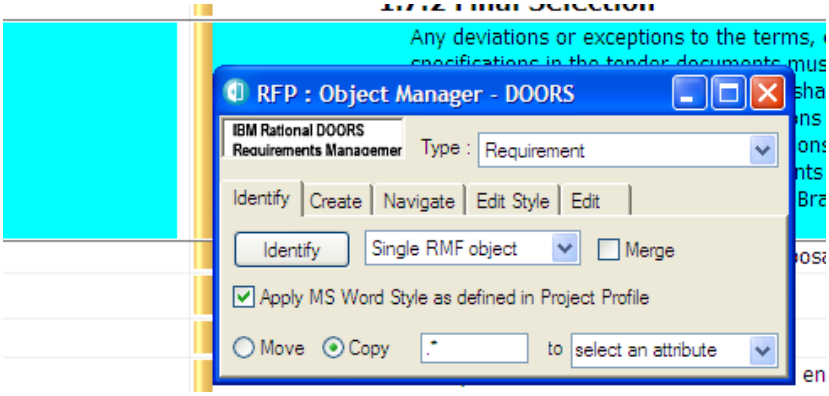


Figure 59 : "Manage Objects" Identify Tab

8.4.2.1 Identifying a selection of objects

The identification tool provides two different modes:

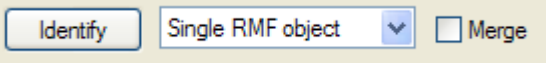
- Single RMF object (default mode)
- Several RMF objects

In "Single RMF object" mode, only one RMF object is created.

In "Several RMF objects" mode, the tool creates as many RMF objects as there are selected objects (except if some objects can't be identified).

Each mode provides its options :

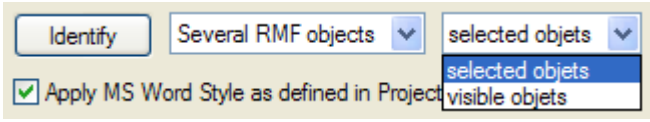
- "Single RMF object" mode options :



Option	*description*
Merge objects ON	The tool merges the selected

	objects and then identify if.
Merge objects OFF (default)	The tool identifiies the first selected object and move the following objects so that they become childs of the newly identified object.

- "Several RMF objects" mode options:



Option	*description*
Visible objects	The tool identifies all the visible objects.
Selected objects (default)	The tool identifiies the selected objects.

8.4.2.2 Applying a MS Word Style

By default the option "Apply MS Word Style as defined in Project Profile" is checked.

Therefore the paragraph style of the new object is set with the value defined in "*Project Profile*" module.

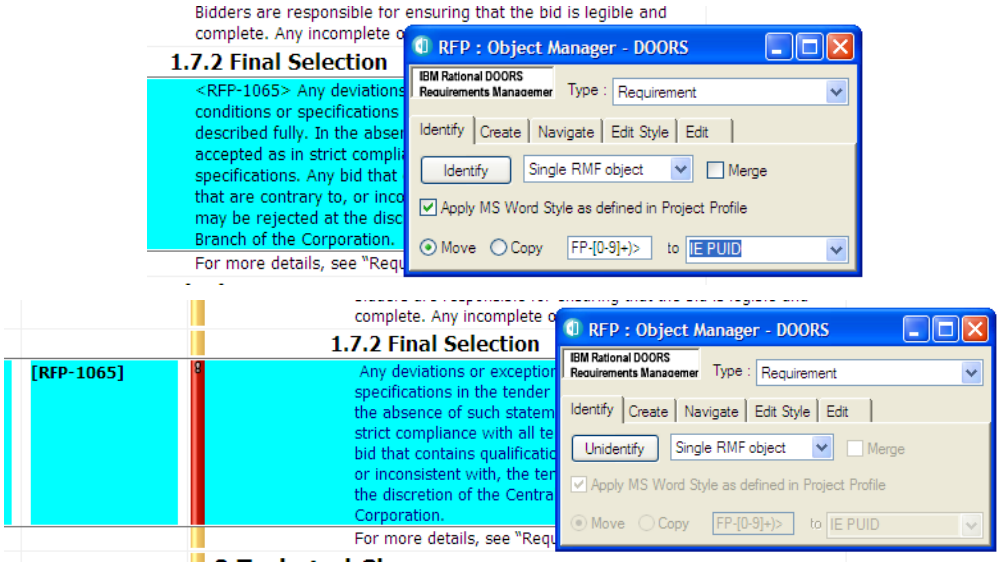
8.4.2.3 Parsing the text of the identified object

The tool allows moving or copying a piece of the "Object Text" attribute value of the identified object, into an attribute to be selected.

The pattern to look for shall be described as a regular expression. Please refer to the DOORS inline help for a comprehensive explanation on the regular expressions syntax.

The tool allows copying or moving only a subset of the pattern to look for. To do so, insert brackets before and after this subset.

For instance, in the example below, the pattern to look for is <SSS-[0-9]+>, but the characters "<" and ">" shall not be moved (or copied). This can be achieved by adding bracket so that the regular expression becomes <(SSS-[0-9]+)>.



8.4.3 Creating new RMF objects

To create an RMF object, select a DOORS object and activate the utility "RMF ->Manage Object". Then choose the "Create" Tab.

Notice that the dialog window stays displayed in front of the screen, allowing repeated object creation with single mouse clicks. Notice also that you cannot create a RMF object within a hierarchy of DOORS 'Normal Text' objects with this utility.

This utility creates a new RMF object after the selected DOORS object:

- at the same level if the current is an object text,
- below if the current is a heading.

The paragraph style of the new object is set with the value defined in "*Project Profile"* module, and the PUID is generated. The new "Object Text" field is initialized with the PUID but it should then be replaced by the desired text.

Once the requirements have been captured and identified as RMF objects, they should be rationalized. Rationalization means splitting up, rephrasing or joining requirements in accordance with the guidelines given in the DOORS booklet "Writing Better Requirements".

Splitting a requirement into two or more requirements is best achieved by "Creating" one or more new requirement objects and cut and pasting the required text.

Joining two requirements can be achieved by cutting and pasting the text from one requirement into another, and then deleting the empty requirement. Alternatively you can also use the same "Manage Object" dialog box that provides a button labelled "Join Objects", which is useful for DOORS objects, prior to them being identified as RMF objects.

In addition, IRDRMFAO allows requirements to be grouped within Functions, which in turn may be grouped within Capabilities, if this aids clarity and understanding. Note that "Requirement", "Function" and "Capability" are just RMF Object Types and that they must all exist at the same DOORS level within a DOORS Heading. The IRDRMFAO utility "Create Links from Object Hierarchy" can be used to aid in the creation of a hierarchy within IRDRMFAO using links such as "is composed of".

Once the requirements have been initially rationalized, the PUID numbering will appear to be randomized or even be absent if editing has been in shared mode. At a convenient time, before releasing the requirements, when "Exclusive Edit" mode is available, they should be renumbered using the IRDRMFAO utility "Renumber Objects PUID", and the module baselined.

8.5 WORKING IN EDIT SHAREABLE MODE

The "Edit Shareable" mode is a DOORS feature allowing several users on a network to simultaneously modify different sections in the same module. If you are working in such a mode, some of the IRDRMFAO commands have a different behaviour than in simple and exclusive "Edit" mode. All of those commands involve allocation or calculation of a PUID, which is impossible in this mode. So the different behaviour is the setting of "IE Object Type" and the "PUID" attribute, which has to be decoupled in "Edit Shareable" mode.

- The "*Manage Objec*t" utility sets the "IE object type" attribute but sets the PUID to a keyword ("TBD",..., or empty) defined in the "*Project Profile*" module (.view "Definition of the object types", attribute "IE Undefined PUID Keyword").
- The "*Renumber Object*" and "*Update Version Attribute*" utilities are disabled in "Edit Shareable" mode.

Creation of identification of a requirement in Shareable mode:

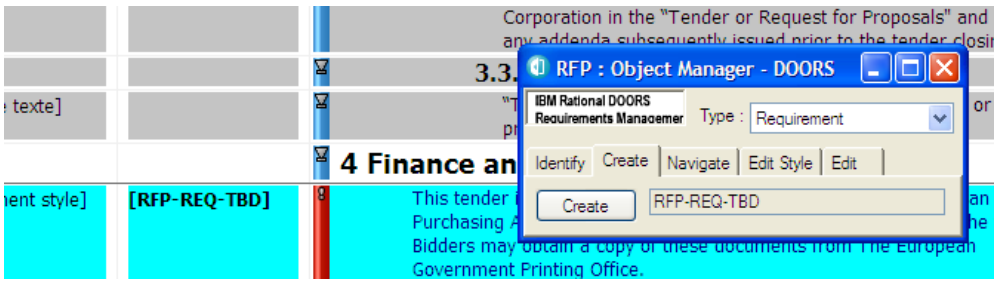


Figure 60 : Shareable mode

Then, to complete the setting of PUID attribute, the module must be open in simple "Edit" mode. The "Renumber object" utility is able to only calculate an appropriate PUID for any objects whose "IE object type" attribute is set and which have a PUID set to the predefined keyword or empty.

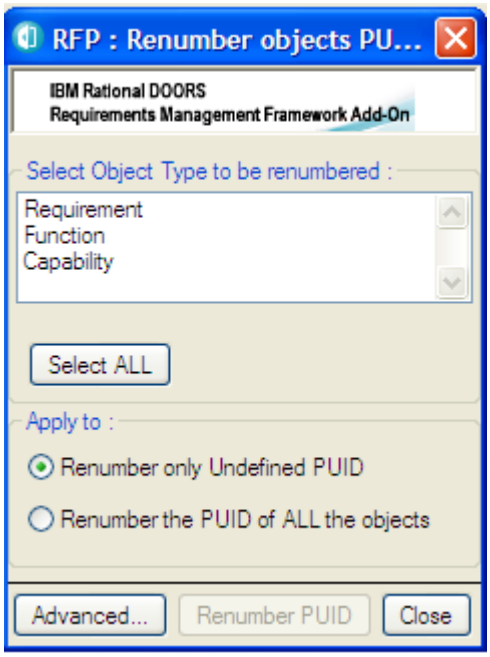


Figure 61 : Renumber utility (simple mode)

By default, the "Renumber" tool is used only to allocate a specific number to the objects having undefined PUID.

You can use it also in advanced mode; to reallocate completely or partially all the existing PUID. For example, if the prefix has been changed and you want to replace the prefix value in all generated identifiers.

You may use it after having created a module in "draft" mode, and want to get all identifiers in the right order. The advanced mode is not recommended because it is always dangerous to modify an existing requirement identifier.

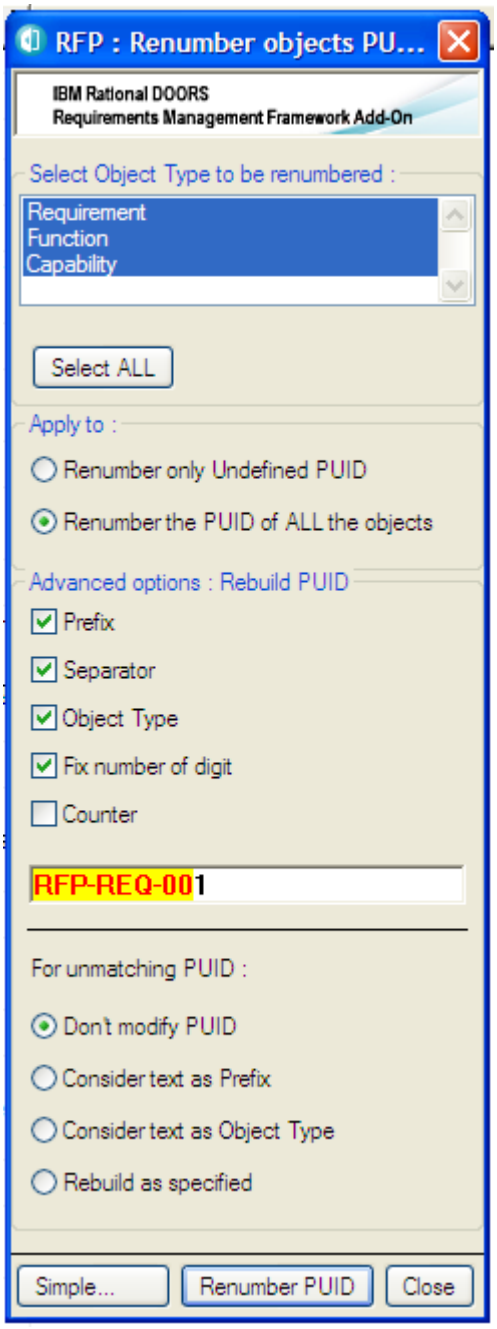


Figure 62 : Renumber utility (advanced mode)

8.6 Defining IRDRMFAO behaviour for New module Types

As described above, IRDRMFAO behaves differently in Requirement type modules from Breakdown structure type modules. This is due to the inheritance setting of the RMF attribute "IE Object Type".

In requirement type modules, the "IE Object Type" is set to inherit its value from its parent, in Breakdown type modules it is not.

To modifying this setting of the attribute "IE Object Type" in a new module type:

- Open the module, activate the menu "Edit->Attribute", select in the list the attribute "IE Object Type" and set or unset the "Inherit value" property.

9 Collecting metrics

Metrics are required to be able to manage your project. RMF contains a dedicated tool allowing to define metrics, and to collect the regularly.

Consult the Dashboard manual to get more information on this functionality.

10 Managing requirement changes

10.1 Introduction

By definition, changes to requirements will occur either outside or inside the IRDRMFAO environment. This section describes how IRDRMFAO assists in the control of both situations.

Changes occurring outside the DOORS environment will in general result in the System Engineer being presented with a later edition of a document which has already been imported and processed within IRDRMFAO, and the changes will probably not be annotated.

Changes occurring inside the DOORS environment will be formally controlled and, if approved, will be applied directly to the reference data.

10.2 Managing changes originating outside doors

10.2.1 INTRODUCTION

This section deals with the problem of source documents that have been already imported into DOORS and analysed within IRDRMFAO, but subsequently modified and up-issued outside DOORS. Typically, such documents come from the customer or other stakeholder.

The IRDRMFAO goal is to reduce as far as possible the work of re-engineering the data by copying the already done analysis on unchanged parts of the document. Unchanged data should not be analysed a second time and existing analysis shall be applied to the new version of the source document module in DOORS.

The process is decomposed into 3 phases:

- The changes to the source document are tracked by comparison mechanisms,
- Then a human analysis completes the comparison,
- The analysis is transferred to the new document module, including links, attributes and views.

This process is not completely formal, because comparison may fail if they are too many differences between two versions of the same document. You should always check the result of the comparison.

To understand the process, assume that we already have a document called “RFP” for example at release V1. This document has already been analysed within a RMF module, as shown in Figure 63. And we’ve just received a new release V2 of the document.

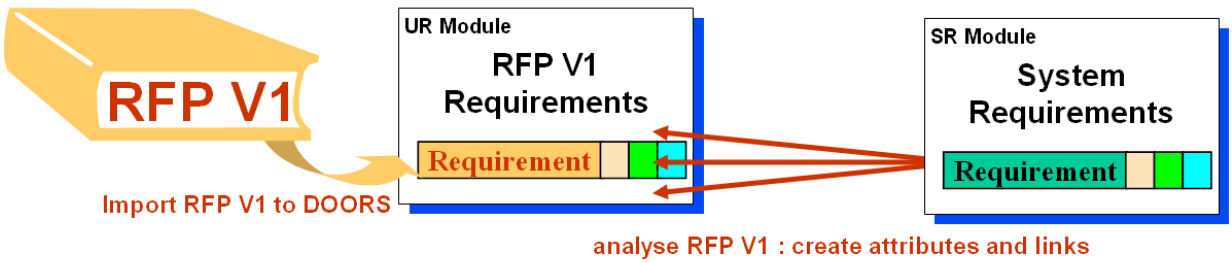


Figure 63 : A source document has been analysed within IRDRMFAO

10.2.2 1st step: import of a new version of the source document

This step uses the standard importer depending on the format of the new version of the source document. The document is imported into a new module.

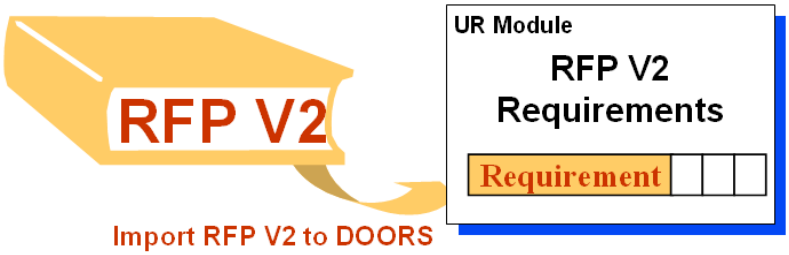


Figure 64 : The new version of the source document is imported

It is important when importing multiple versions of a document that the import process is identical. If not, the comparison will be more complicated.

10.2.3 2nd step: comparison of the two versions of the source document

Different comparison algorithm are available.

The RMF tool should be better used than the standard DOORS comparison tool, because the DOORS tool is designed to produce results for human analysis.

The IRDRMFAO comparison tool includes two different algorithms, the classic and hierarchical algorithms. Use the first one, and if the result is not good try the second one.

10.2.3.1 GUI and setup

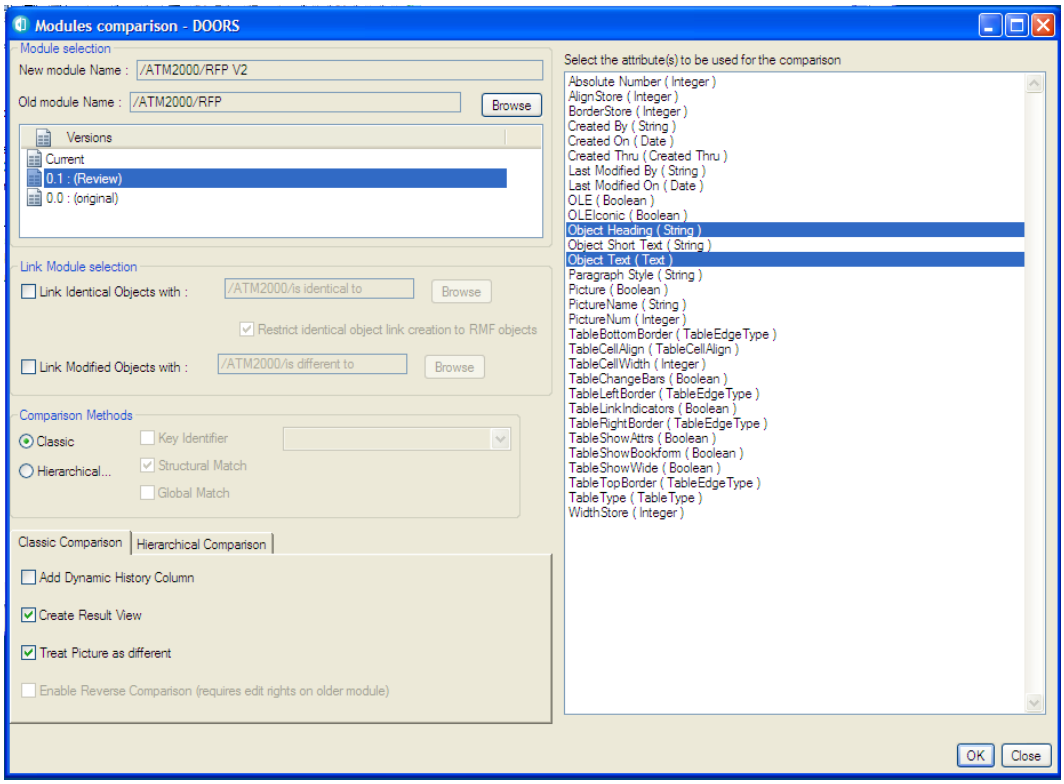


Figure 65 : "Compare Modules" Graphic User Interface

The main features of the "Compare Modules" function are:

- Compare a module current version to any module current version or baseline.
- Create links between the two compared modules to track the changed or unchanged objects. These links are created **from *the New module* to the *Old one***, whatever the chosen algorithm.

There is an important difference between running the tool in the New→Old direction, and in the reverse direction (Old→New). If running in the direction New→Old, the tool generates some new attributes and views in the New module, to which it must therefore have write access. It will be incapable of displaying any objects unique to the Old module, that is to say it cannot display any objects that were deleted between the old and new issues of the document.

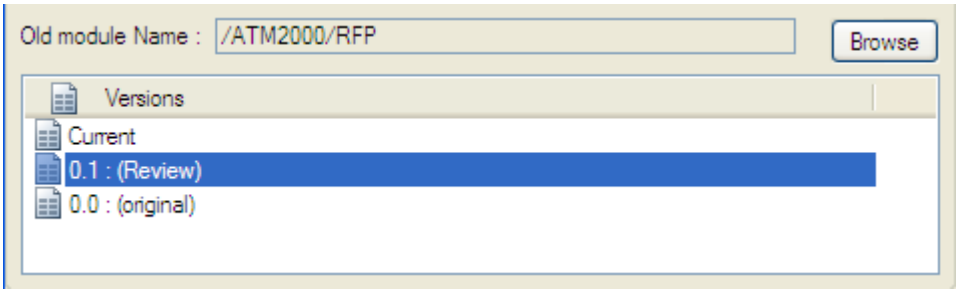
To run in the direction New→Old, open the new module (RFP V2), and launch the "***Compare Modules***" command in RMF menu..

To run in the direction Old→New, open the old module (RFP), and launch the "***Compare Modules***" compare in RMF menu.

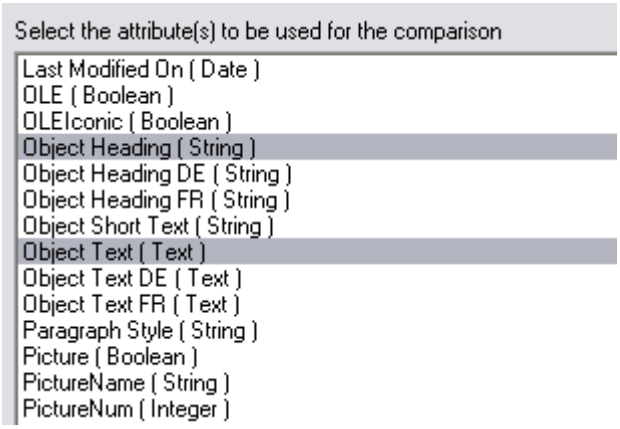
The various parameters to be defined are:

- **Select the new module**. Use the RMF - "Compare Modules" from the RFP V2 module; then the New Module Name will automatically be "RFP V2", and there is no way to change it.

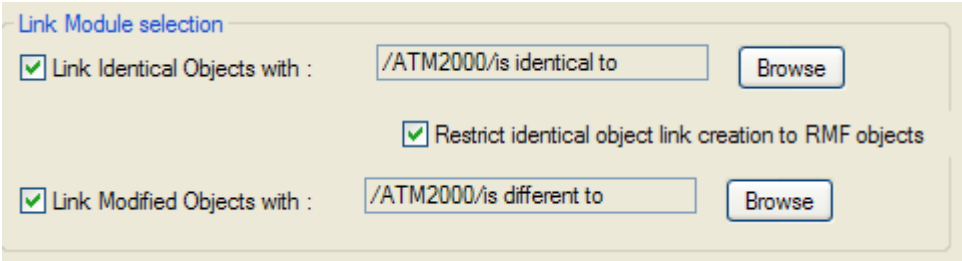
- **Select the old module and its version**. Use the browse button to select the old module, and use the list to select the version.



- **Select the attributes to be used for the comparison**. The default selection is "Object Heading" and "Object Text".



- **Select Link Module for identical objects**. It is the link module to be used to link the **identical** objects between the two modules. If no link module is selected, the identical objects won't be linked together. Default value is "is identical to"
- **Select Link Module for modified objects**. It is the link module to be used to link the **similar** objects between the two modules. If no link module is selected, the similar objects won't be linked together. Default value is "is different to"



"Identical Objects" and "Modified Objects" links are created from the new module to the old one, and are stored as specified.

The following figure shows how links are created:

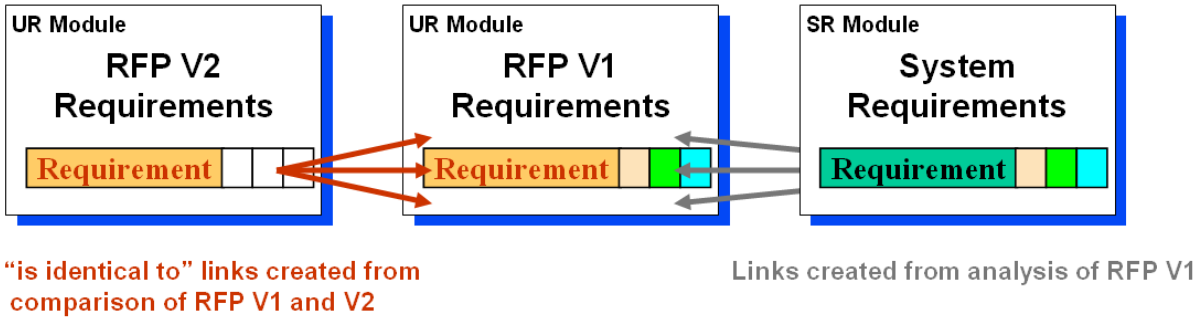
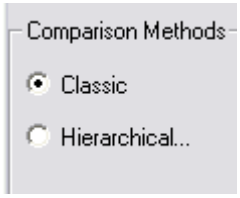


Figure 66 : Links created between V1 and V2

- **Select the algorithm to be used.** Select the "Classic" or the "Hierarchical" comparison method using the "Comparison Methods" radio buttons:



10.2.3.2 Pre-processing verifications

Before trying to process the comparison, tests are made regarding the linkset pairing definition.

- If no comparison link creation is required no tests are made.
- If comparison link creation is required:
  - When the linkset pairing ***is not enforced*** then links will be created.
  - When the linkset paring ***is enforced***:
    - Either **you have the "administrate right"** on the folder containing the new module; then the enforcement will be cleared and restored. The action will be notified in the log window.

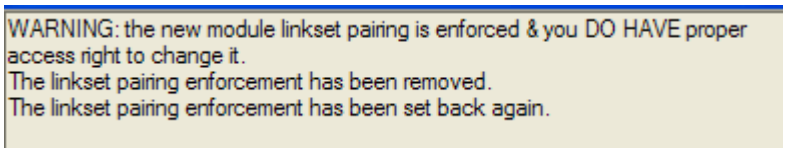
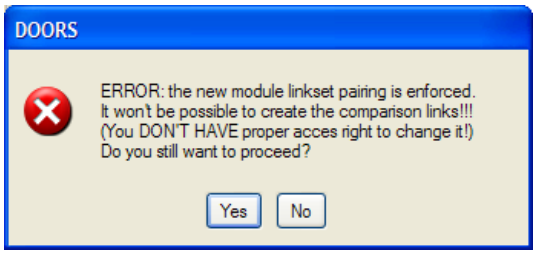


Figure 67 : pre-processing log example (temporary linkset pairing unenforcement)

    - Or **you don't have the "administrate right"** on the folder containing the new module; then the following window is displayed, which allows you to go on (in which case errors will occur) or not to launch the comparison. The choices made will be logged.



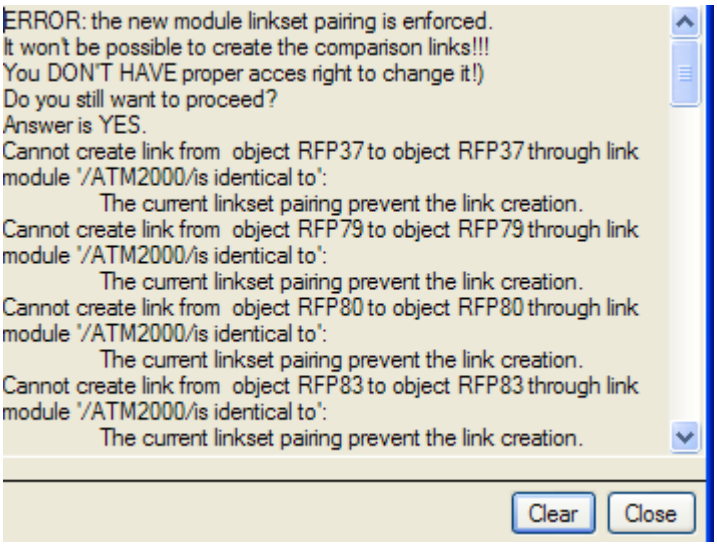


Figure 68 : pre-processing error window and log example (linkset pairing prevents link creation)

10.2.3.3 Processing verifications and log window

If errors occur during the link creation, you'll be asked to skip the current single error, or to skip all the errors, or to cancel the comparison.

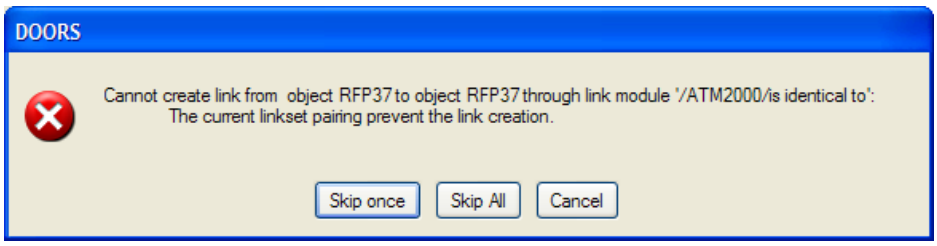


Figure 69 : processing error confirmation window example

A log window is displayed at the end of the processing to keep trace of all the errors that occurred:

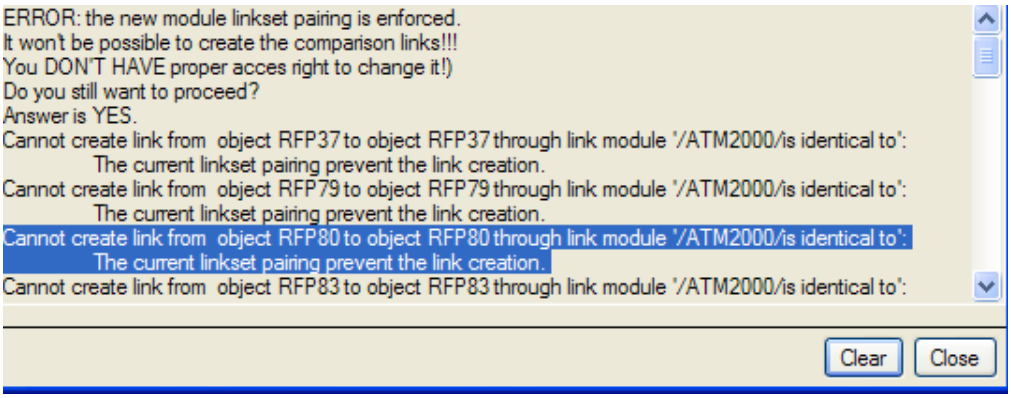
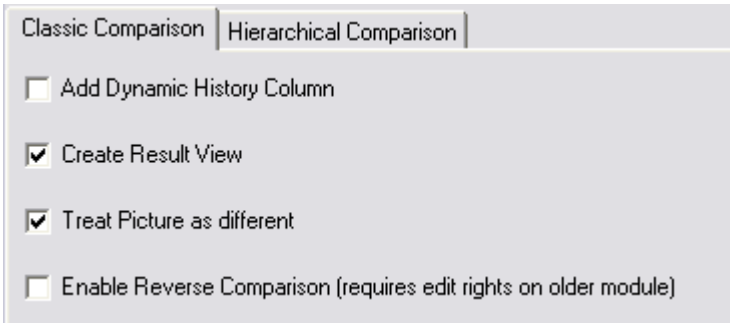


Figure 70 : processing log window example

10.2.3.4 Classic algorithm

10.2.3.4.1 Use and Parameters

The specific parameters for this algorithm are the ones displayed in the "Classic Comparison" tab.



- **Add Dynamic History Column.** If checked, a dynamic DXL column will be added to the current view (in the far right-end position) and shall display the differences between the old and the new module. Standard rules are used to mark the differences relative to the non-current module. Deleted text is struck through and inserted text is underlined. For example, if some text has been added to the new module it is underlined when the new module is current (i.e. Compare has been run in the direction New→Old). On the other hand, if the old module is current (i.e. Reverse Compare has been run), this same text is struck through.

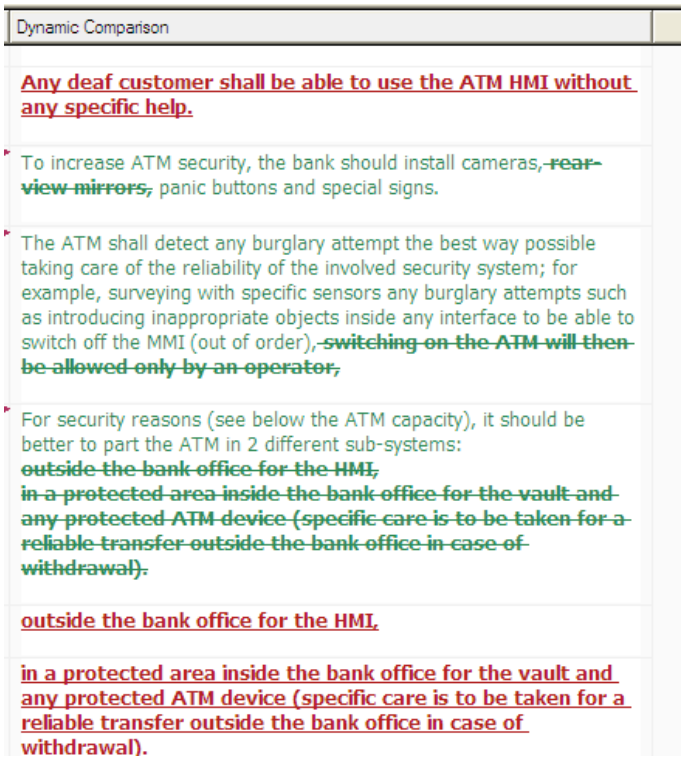


Figure 71 : Example of Dynamic comparison column for classic comparison

- **Create Result View.** If checked, a specific view including the DXL Column "Dynamic History" is created and saved. A filter will automatically be set to show only modified, new or deleted objects.

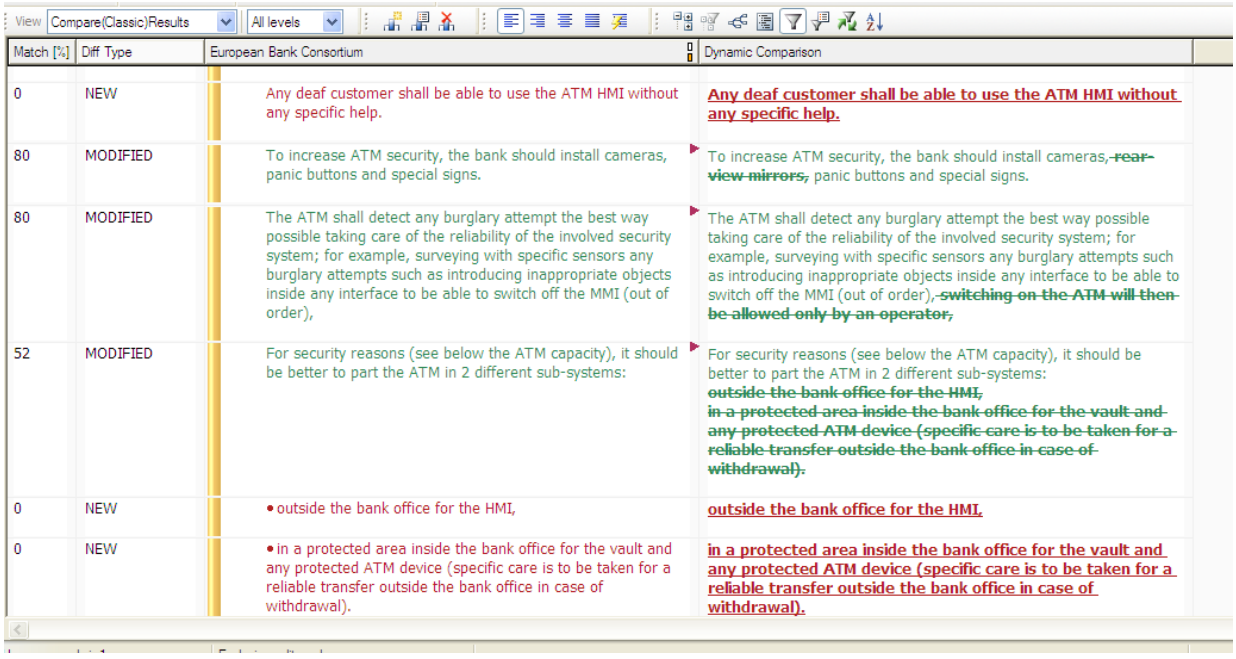


Figure 72 : Example of a result view for classic comparison

- **Treat Picture as different.** As the algorithm is not able to compare pictures, it's possible to force pictures to be considered as different, and a human analysis shall be done to set the result attribute to identical or modified regarding the applied modifications to the pictures.
- **Enable Reverse Comparison.**

To be able to see deleted objects within a result view, the Classic method offers the 'reverse compare' option – when run in the direction Old→New, deleted objects will be displayed, but conversely objects added to the new module will not.

10.2.3.4.2 Final result window

At the end of the comparison a log window displays some figures that can be exported to a text file.

10.2.3.4.2.1 Standard (i.e. non reverse) comparison

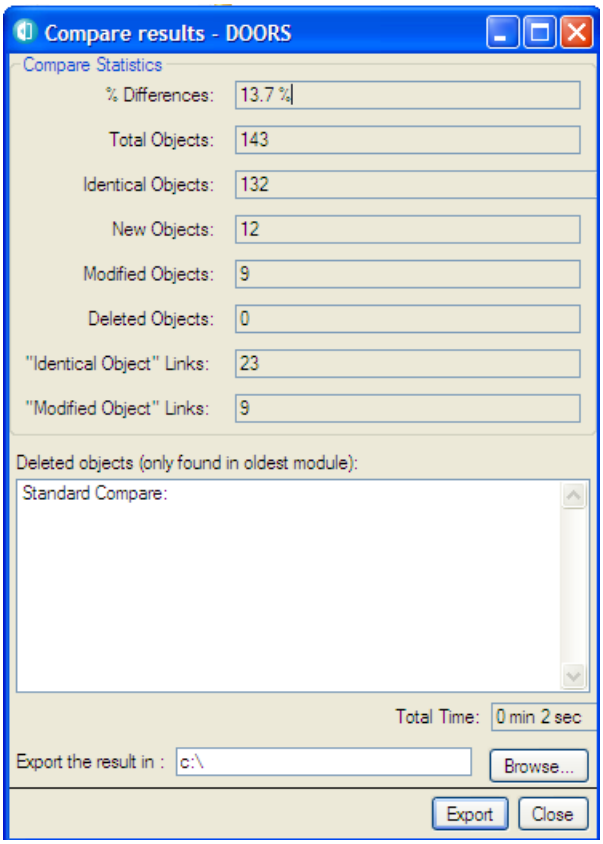


Figure 73 : Final result window for (classic) non reverse comparison

10.2.3.4.2.2 Reverse comparison

In this case, the final result window is slightly different.

For each figure, the result for the standard comparison and the one for the reverse comparison are displayed separated with a slash character. The first value is for the standard comparison and the second one is for the reverse comparison.

As for the "Deleted objects" list, you have a Standard Compare set of deleted objects AND a "Reverse Compare" set of deleted objects.

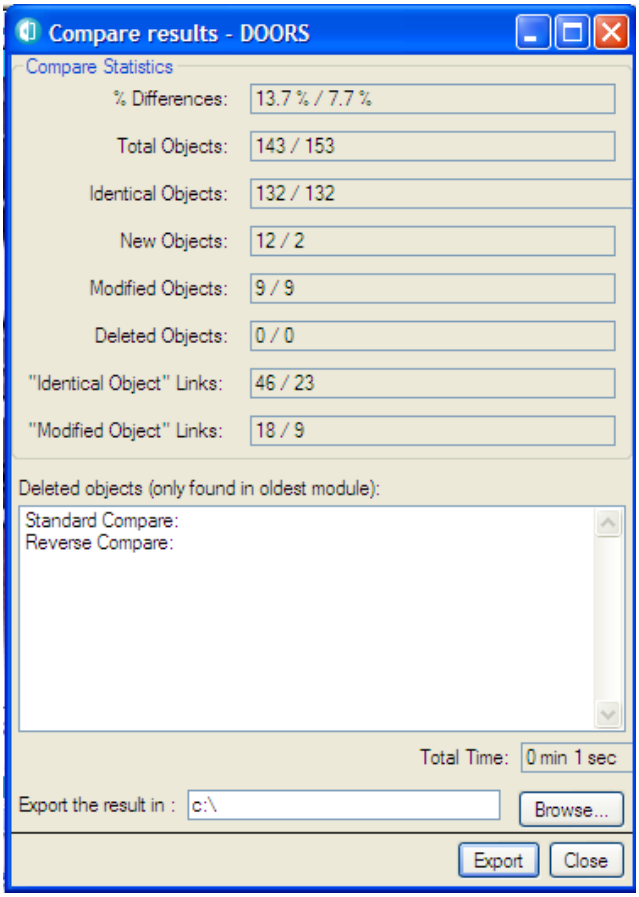


Figure 74 : Final result window for (classic) reverse comparison

10.2.3.5 Hierarchical algorithm

In comparison with the "Classic" algorithm, this one can:

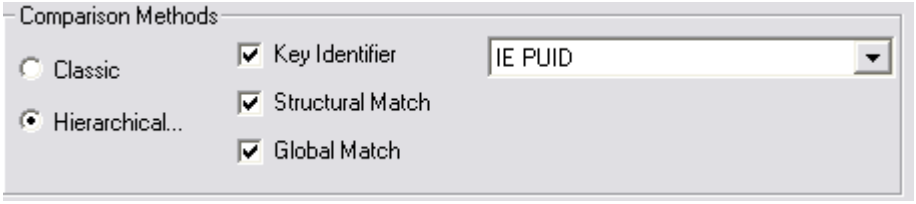
- handle requirements in a special way: as requirements have an identifier which should remain unchanged, the algorithm can look for requirements knowing their identifying attribute,
- perform a comparison based on the structure of the modules (for non-requirement objects). This option dramatically speeds up the comparison, but requires that the structure of the two modules compared is rather similar.

This tool cannot:

- compare graphical objects and OLE objects (In general this is not a restriction insofar as these objects do not contain requirements. If they do, they should be treated manually).

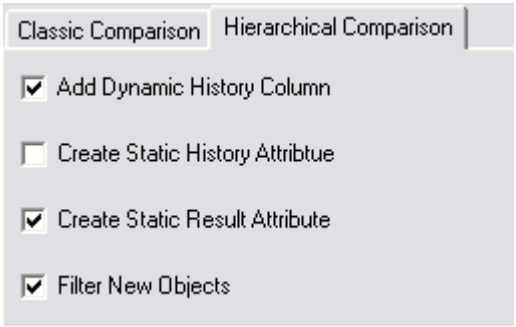
10.2.3.5.1 Use and Parameters

For this algorithm you can mingle several comparison methods: key identifier, structural match and global match.



- **Key Identifier.** Activation or not of the research algorithm using an identifier attribute. Simply check the "Key Identifier" box to enable this functionality and select the attribute (only integer and string attribute are listed in this choice list). Whenever this attribute is not empty, it is supposed to be a single and stable identifier of the current object, and can be used to find the equivalent object in the other document. Default value is "IE PUID".
- **Structural Match.** Activation or not of the structural research algorithm. This research is based upon the structure of the document. Two objects are equivalent if their parents are equivalent and if the values of their compared attributes are equal or similar. This option is active by default.
- **Global Match.** Activation or not of the full research algorithm. By default it is not selected. If the difference between the two modules is light, and in particular if there is no difference in structure, this additional stage is not necessary. If the number of objects remaining after the first stage is important (more than one thousand), then this second stage can be time consuming.

Other parameters are available in the "Classic Comparison" tab. As soon as one of the following parameter is set "on", a view named "Compare(Hierarchical)Result" is created and automatically saved, if ever you ask it to be at the end of the comparison.



- **Add Dynamic History**. If activated a DXL column is created in the current view. It shows the differences between the old and the new objects in a dynamic way. The "**Old**" attributes are created and initialized with the values found in the old module. This column is automatically updated whenever a modification is made in the text of the new module.

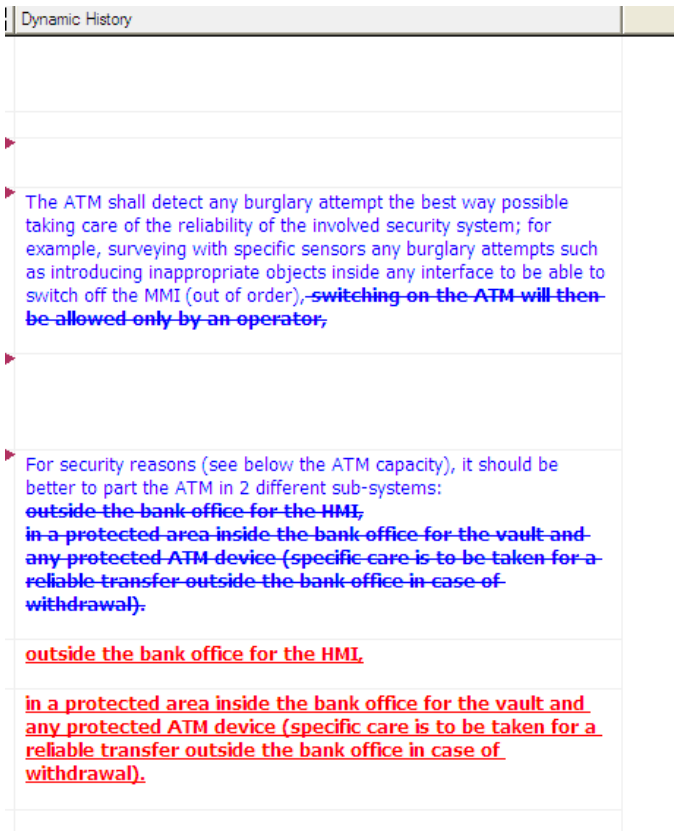


Figure 75 : Example of "dynamic history column" (hierarchical method)

- **Create Static History Attribute**. Same as last parameter, but the difference is computed only once and recorded into the attribute "**Static Difference"**.

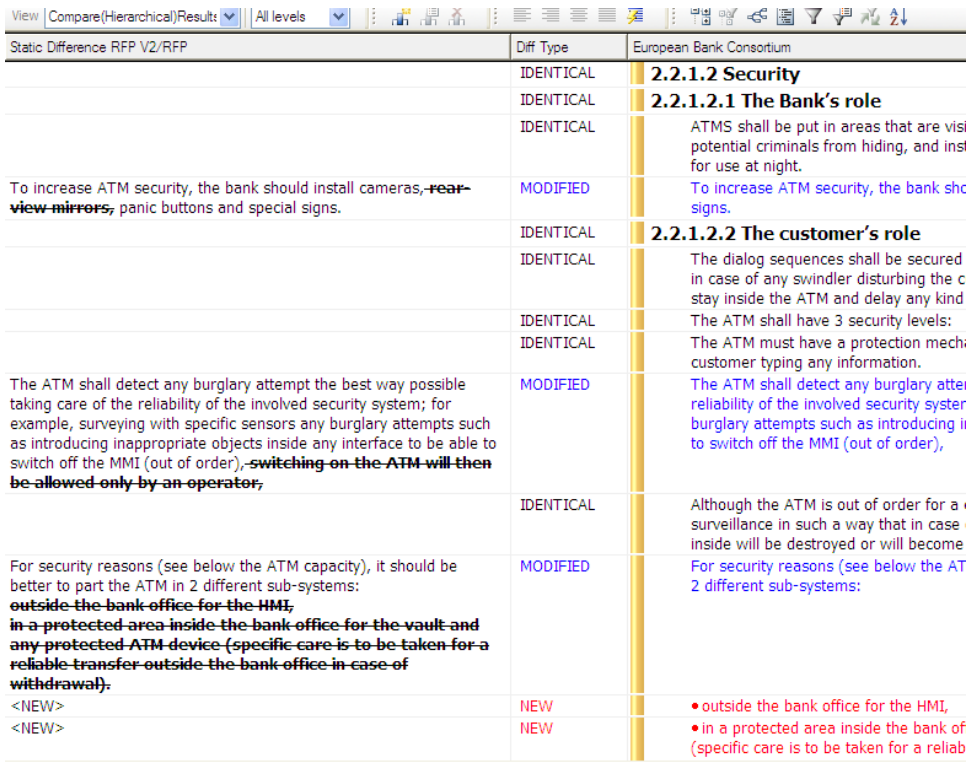


Figure 76 : Example of "static history attribute" (hierarchical method)

- **Create Static Result Attribute**. If activated, the attribute "**CompareHierarchicalResult"** is created and records for each object whenever it is changed, unchanged or new. It is displayed in the "Diff Type" column.

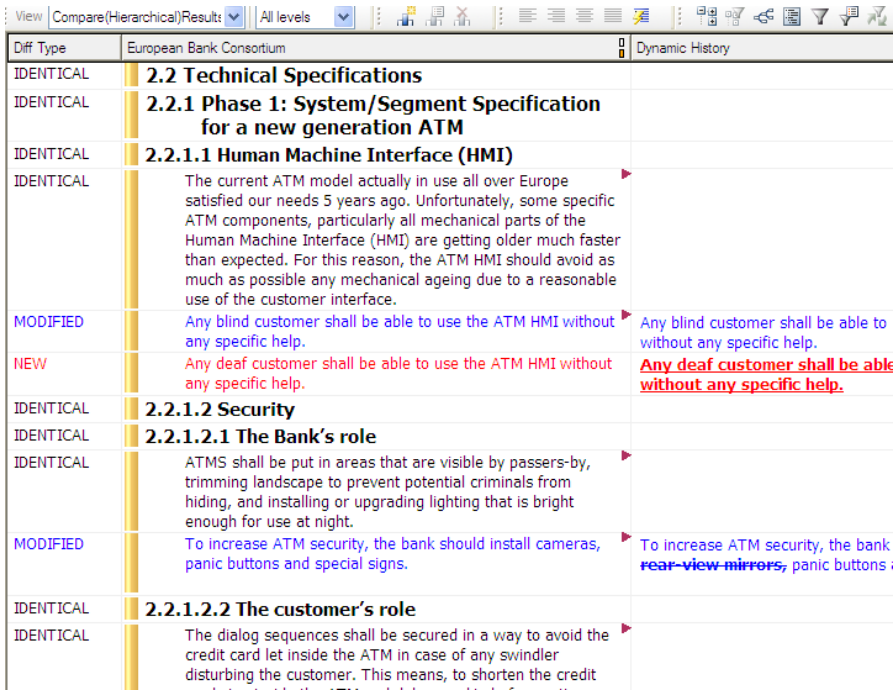


Figure 77 : Example of "static result attribute" (hierarchical method)

- **Filter new objects.** If activated only the objects without equivalents will show in the two modules after the comparison (objects destroyed in the old version and created in the new).

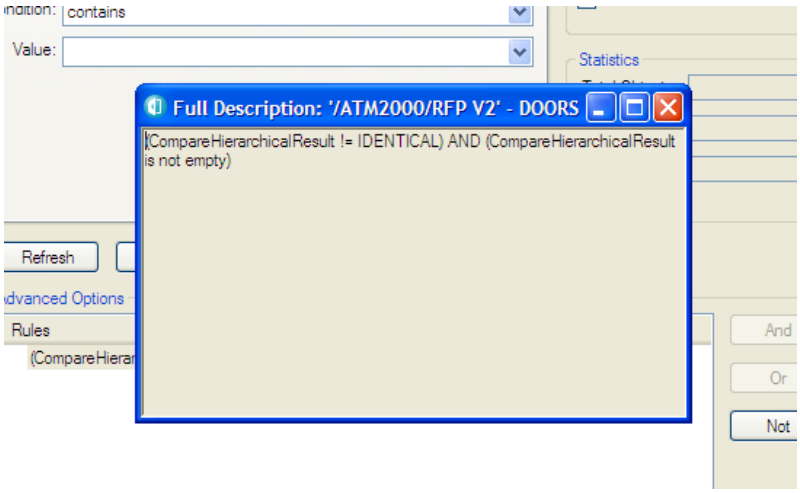


Figure 78 : Filter description (hierarchical method)

The color of the objects has the following meaning:

- The black-colored objects are unchanged and may have outgoing links stored in the "is identical to" link module (or similar).
- The blue-colored objects are changed as compared to the object of the other module and may have outgoing links stored in the "is different to" link module (or similar).
- The red-colored objects are new and won't have newly created link.

Views can be created to display the results of the comparison by using:

- The links, (and possibly traceability columns)
- The attribute "**CompareHierarchicalResult**" (whose values are IDENTICAL, MODIFIED or NEW)
- The attribute Static History, which shows the differences in the text between the new and the old versions, using various styles:
  - Strike-through for deleted characters,
  - Underlined for additional characters
- The "**Dynamic History**" column (similar to the "**Static Difference**" attribute) and the "Old" attributes.

10.2.3.5.2 Save confirmation window

At the very end of the comparison, a confirmation window is displayed:

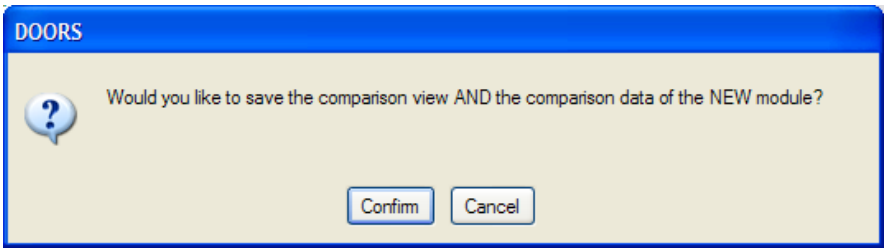


Figure 79 : Final confirmation window for a non reverse comparison

In this case it means: "Do you want to save the comparison view in the new module?" and "Do you want to save the new module?"

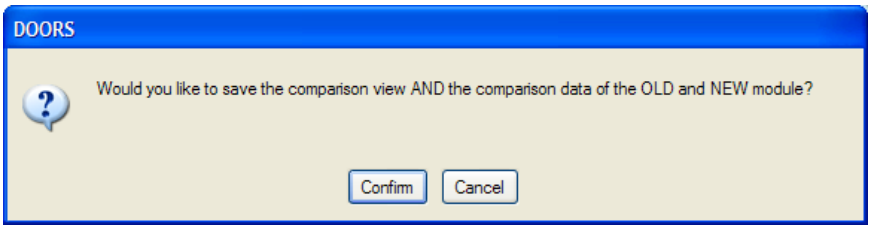


Figure 80 : Final confirmation window for a reverse comparison

In this case it means: "Do you want to save the comparison view in the old and in the new module?" and "Do you want to save the old and the new module?"

Be careful, if you decide not to save (i.e. click on the "Cancel" button), just avoid saving the views without the related modules or else the views will generate DXL errors, for views will miss useful attributes.

10.2.4 3rd step: Human check

At this phase, just before transfer analysis, if needed the user may create additional "is identical to" or "is different to" links in order to link objects he wants to be treated as unchanged (even if the text has been changed, due to spelling, paragraph styles, ...).

10.2.5 4th step: Transfer analysis

This step uses the RMF tool "**Transfer Analysis**". .

It's possible to transfer attribute type, definitions and values, links and views. As for views and attribute definition, the behaviour is now based upon the synchronization mechanism.

A module processed with "Transfer Analysis" don't need to be tagged, because the definitions of attributes and views are duplicated from the previous version.

Notice that composite "objects" would be restructured under certain conditions:

- The old module is a RMF module
- The old "object" is a RMF composite "object"
- All the objects part of the composite "object" shall have comparison links
- All the linked new objects shall be contiguous and in the same relative position.

10.2.5.1 Graphic User interface

10.2.5.1.1 Simple GUI (i.e. collapsed advanced panel)

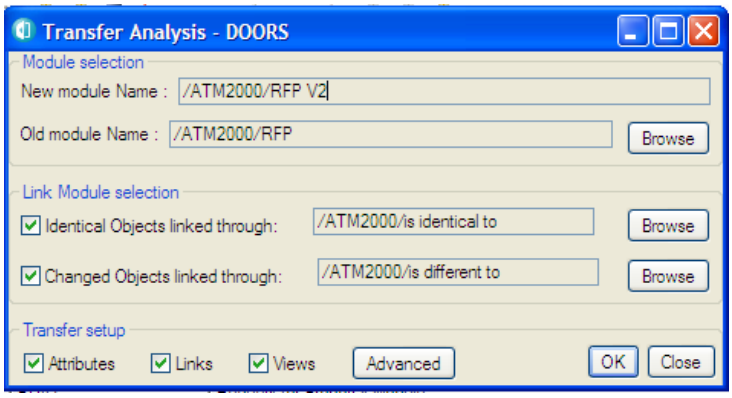


Figure 81 : Simple Graphic User Interface for Transfer Analysis

10.2.5.1.2 Advanced GUI i.e. (including the advanced panel)

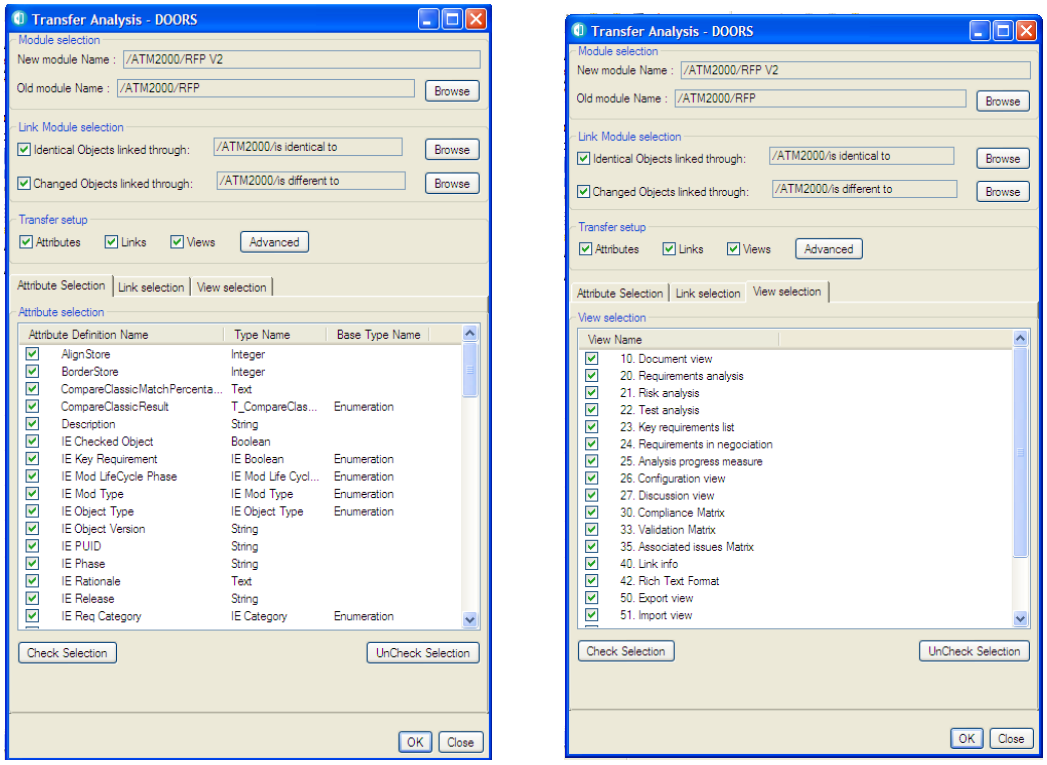


Figure 82 : Advanced Graphic User Interface for Transfer Analysis: Attribute & View selection

Notice that system attributes that are read only (for instance the "Absolute Number" attribute) in the target module are not proposed.

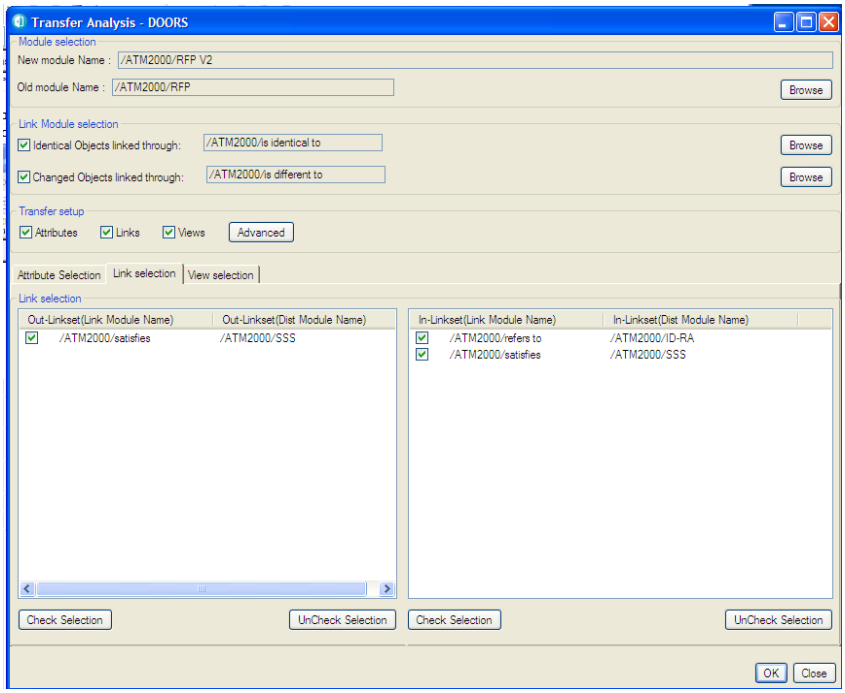


Figure 83 : Advanced Graphic User Interface for Transfer Analysis: Linkset selection

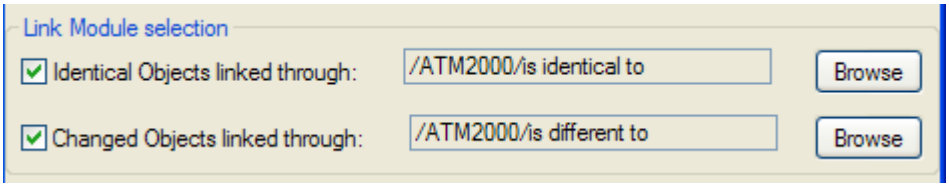
10.2.5.2 Use and Parameters

The "Transfer"parameters are hereafter described.

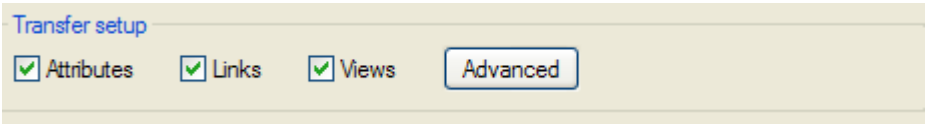
- **Select the new module**. Use the RMF - "Compare Modules" from the RFP V2 module; then the New Module Name will automatically be "RFP V2", and there is no way to change it.

- **Select the old module**. Use the browse button to select the old module.

- **Select Link Modules for identical objects**. It is the link module to be used to find the old object information to be transferred, for new identical objects.
- **Select Link Module for modified objects**. It is the link module to be used to find the old object information to be transferred, for new similar objects.



- **Transfer setup**. Just check the boxes in order to define what shall be transferred from the old module to the new.



- **Advanced Transfer setup**. Specify amongst all the possible data what data you want to transfer. For instance you can transfer a subset of all the available attributes, only specific links (chosen amongst linksets), and a subset of all the available views. The lists of attributes, links and views are built in order to propose the old module data. See advanced GUI screen shots.

Be cautious using these attributes for it may result in inconsistent configuration, especially between attributes and views.

So, in our example, it copies to RFP V2 the attributes and the links of RFP V1 unchanged objects in RFP V2, as shown in Figure 84. Note that if they exist, both incoming and outgoing links can be copied.

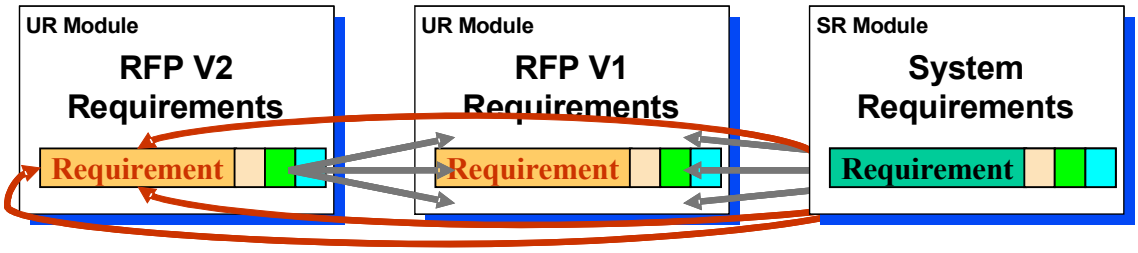


Figure 85 : Attributes and Links copied from comparison of RFP V1 and V2

Notice that the default selections are set as followed:

- **Attribtues:**
  - If the *old module is NOT RMF* and *the new one is NOT RMF*
    - All the attributes are selected except "Object Heading", "Object Text", and "Paragraph Style".
  - If the *old module is NOT RMF* and the *new one is RMF*
    - All the attributes are selected except "Object Heading", "Object Text", and "Paragraph Style"
  - If the *old module is RMF* and the *new one is NOT RMF*
    - All the attributes are selected except "Object Heading", "Object Text", "Paragraph Style", RCM and PFM attributes (the "IE PUID" and "IE Object type" attributes are selected).
  - If the *old module is RMF* and the *new one is RMF*
    - All the attributes are selected except "Object Heading", "Object Text", "Paragraph Style", RCM and PFM attributes, "IE PUID" and "IE Object type" attributes.
- **Linksets**
  - The default selection includes all the linksets except the comparison ones.
- **View**
  - The default selection includes all the views.

Notice too, that default selections can be changed through a modification of user callout functions. See the comments in the code to know how to use them.

The functions are located at:

`"$IRDRMFAO\lib\dxl\addins\IRDRMFAO\brandnewtransfer\includes\usercallouts.inc"`

The functions are :

- ***processDefaultAttributeSelection***: function to be modified (see the end of the function for an example) to change the default attribute selection.
- ***processDefaultLinksetSelection***: function to be modified (see the end of the function for an example) to change the default linkset selection.
- ***processDefaultViewSelection***: function to be modified (see the end of the function for an example) to change the default view selection.

processCheckBeforeExecution: function to be modified (see the end of the function for an example) to add a specific check before the transfer execution.

The "Paragraph Style" attribute is copied in a specific way:

- If the attribute is transferred from a RMF object, the local value is not modified (no transfer)
- If the attribute is transferred from a non RMF object, the local value is replaced by the transferred value.

Notice the following contraint:

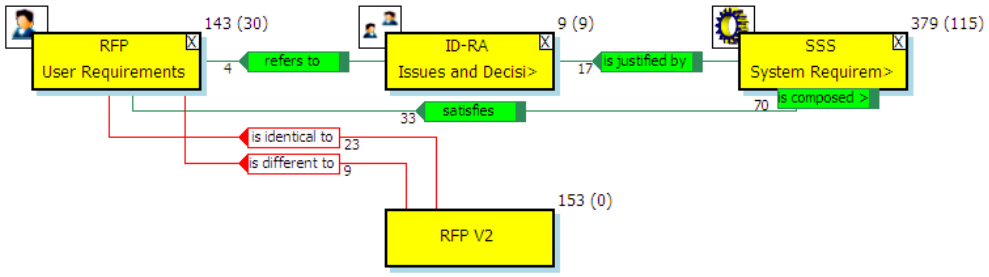
To transfer incoming links, you need **write access rights** to the **source module(s)** of those links**.**

10.2.5.3 Pre-processing verifications and log window

Before trying to process the transfer, tests are made regarding the linkset pairing definition.

- If no link transfer is required no test are made.
- If link transfer is required, the linkset pairing will be checked for all the necessary couples of modules. Interesting events and modifications will be logged in a log window.
    - Generally, the transfer script will try to update the linkset pairing so that to reproduce the "same" new module linkset pairing as the old module one. It can be an update of:
        - either the new module linkset pairing itself in the case of an outgoing link transfer
        - or a reference to the new module in another module linkset pairing in the case of incoming link transfer
    - When the linkset pairing ***is not enforced***
        - Either **you have the "administrate right"** on the proper folder; then the linkset pairing will be updated only if ever a linkset pairing was defined for the old module. If not, there will just be a link transfer.
        - Or **you don't have the "administrate right**" on the proper folder. You will then be informed of the situation in the log window. The links will be transferred anyway.
    - When the linkset paring ***is enforced***:
        - Either **you have the "administrate right"** on the proper folder. In this case the linkset pairing will be updated only if ever a linkset pairing was defined for the old module. If not, the enforcement will be temporarily removed.
        - Or **you don't have the "administrate right**" on the proper folder. In this case the link won't be transferred but you can go on still. The Errors will be logged (see § 10.2.5.4).

Example:



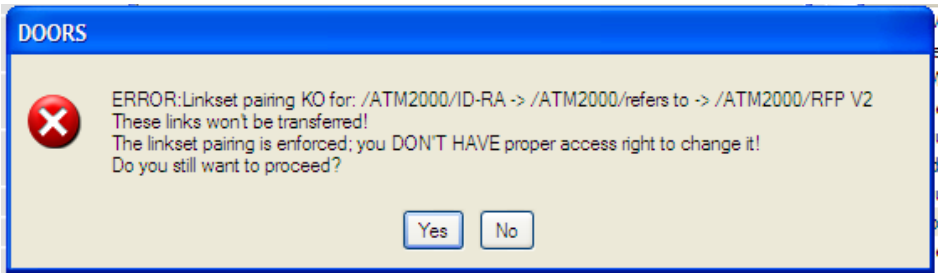


Figure 86 : Linkset pairing error example

10.2.5.4 Processing verifications and log window

If errors occur during the attribute transfer, the log window will list the errors.

If errors occur during the link creation, you'll be asked to skip the single current error, or to skip all the errors, or to cancel the transfer.

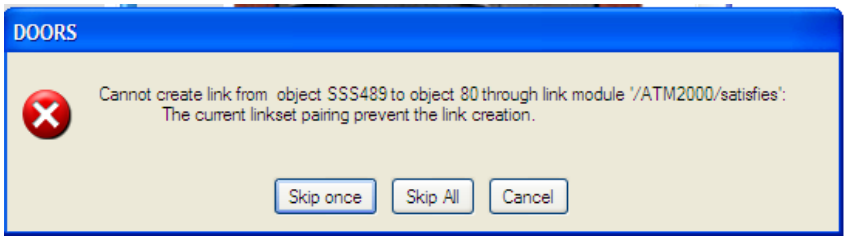


Figure 87 : processing error window example (linkset pairing prevent link transfer)

A log window is displayed at the end of the processing to keep trace of all the errors that occurred, including informations from the pre-processing verifications:.

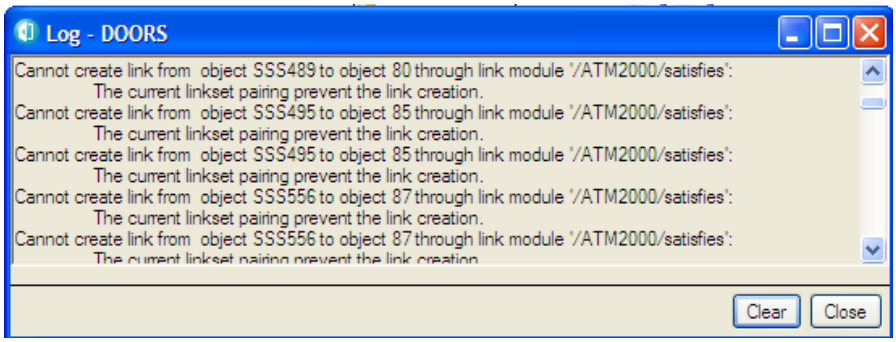


Figure 88 : processing log window example (linkset pairing prevent link transfer)

10.2.6 5th step: optional deletion of the older version of the source document

Now RFP V1 could be deleted, or archived (in order to keep trace of the history), because it is no longer necessary for the traceability from System Requirements to new version of RFP V2. Its continued presence in the list of modules in a project becomes a source of potential confusion.

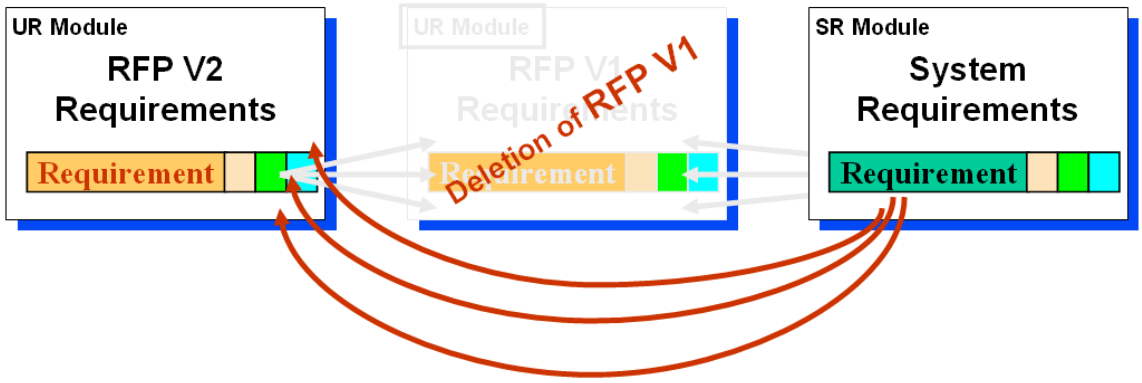


Figure 89 : Deletion of old version of the DOORS module.

NB: you can use the Explorer to do this easily.

10.3 Managing change within DOORS

Within a DOORS environment, change is best managed on a change by change basis, rather than having to first find and then process a batch of changes as described in the previous section. Change proposals will originate from any stakeholder or system engineer, but only those approved by the relevant authorities should be applied.

Setting a Change Control process is generally useful on a stable system, when the engineering work is practically terminated, and you want now trace and control every single modification from one specific state.

Different solutions may be deployed to set a Change Control process on DOORS data:

- DOORS Change Proposal System. It is a function built-in in DOORS, allowing to capture any evolution request, and with a simple workflow to accept or cancel requests. Accepted requests will be automatically applied. Requests are saved into dedicated modules.
- DOORS/Change Integration. Change is a generic Change management tool, designed to be flexible and open. The integration allows you to manage Change Requests on a Web Interface or in DOORS, and to collect any modification in a set of DOORS data to associated them with a Change Request. The request may be accepted or cancelled.

With these two mecanisms, the different states of the system should be saved into DOORS baselines or baseline sets.

An alternative solution is the deployment of one component of RMF, RCM. RCM is a basic layer for the Product Family Management functionalities, but it can work independently.

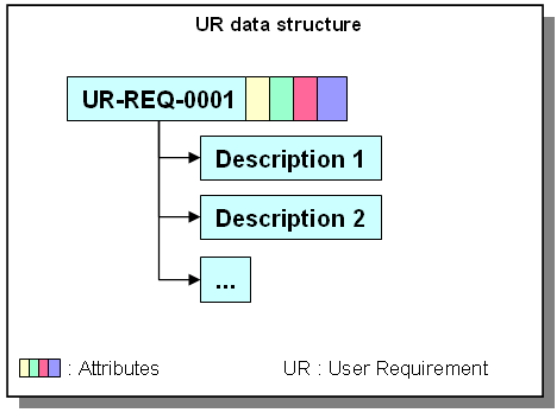
RCM manages version of objects without requiring the creation of baselines, to capture the different configurations of versions. RCM contains also a minimal Change Control management process, allowing to create Change Requests, to associate modification of objects with Change Requests and to accept or cancel Change Requests. RCM is described into the RCM manual.

Appendix A. User Requirements Module Description Form

This module description form contains information about DOORS formal module of type "User Requirements".

The "User Requirements" Module contains RMF objects of type "Requirement". The data structure for each such object is a tree. The root DOORS object is the User Requirement itself. It carries all the attributes applying to the User Requirement, it is the only DOORS object in the tree, which can be identified as a RMF object and gain a PUID. All links to and from the object should be attached to the root object

The leaves of the tree are DOORS objects which inherit the attribute "IE Object Type", which for a User Requirements, has the value "Requirement". The leaf DOORS objects should only be used for descriptive or explanatory material, and all of their RMF attributes should be set to ‘blank’. Leaf objects cannot be assigned a PUID, and should not be linked using standard RMF linksetsThe enumerated lists shown in the following tables are the defaults supplied with the delivered IRDRMFAO. They should all be customised to be appropriate to the individual project



Typical setup/process for the "User Requirements" Module

Assumption : Originating Document is available from WORD. It could be any other format readable by DOORS (see import format supported).

1) Create a new empty "User Requirements" module, by restoring (using file>restore>module in the Database Window) the file UR.dma.

1) From WORD, export the document to DOORS, by clicking the "Export To Doors" icon in the WORD tool bar,

2) Within DOORS, select paragraphs that represent a user requirement and use the IRDRMFAO command "Manage Objects” to identify Requirements,

4) For each user requirement identified, fill the attributes using the DOORS views.

Module attributes of User Requirements module

Attribute name	Type	Value	Description
IE Mod Type	enum	User Requirements	Type of the module, always "User Requirements" in this case. For the whole list of values, see the "Project Profile" Module Description.
IE Mod LifeCycle Phase	enum	Feasibility Study Model Simulation Demonstrator Prototype Full Development Production Operation & Support	Represents the life cycle phase of the project to which the requirements identified in the module apply. .
IE Requirement Number	Integer		An internal counter used by IRDRMFAO, do not edit.
IE Release	string	see description	Stores the IRDRMFAO release which has been used to create this module.

Object attributes of User Requirements module

These attributes apply to the requirements (i.e. "IE Object Type" attribute is "Requirement")

Attribute name	Type	Value	Description
IE Object Type	enum	Requirement	Type of the object, always "Requirement" in this case, or null. For the whole list of values, see the "Project Profile" Module Description.
IE PUID	string	see description	Project Unique ID. This attribute is automatically set with the following concatenation rule : [module prefix]-[requirement prefix]-[number] - the [module prefix] is obtained from the DOORS module attribute "Module Prefix", - the [requirement prefix] is obtained from the DOORS attribute "IE Object Prefix" (see the "Project Profile" Module Description), - the [number] is incremented by one every new requirement.
IE Req Priority	enum	Low Medium High	Represents the level of priority the customer has set to the requirement.
IE Req Compliance	enum	Not Met Partially Met Met	Represents the level of compliance obtained with the System Requirements (see "System Requirements" Module Description) against the customer requirement.
IE Req Flexibility	enum	Low Medium High	Represents the level of flexibility the customer could have when negotiating the requirement.

Attribute name	Type	Value	Description
IE Req Category	Enum multival	Mission Operational Objective Operational Scenario Contract Management Functional - Function - Data - Behavior Performance Dependability - Reliability - Availability - Maintainability - Security - Safety Constraints - Cost - Schedule - Architecture - IV&V - Development - Production - Delivery - Ergonomy - Installation	Allow categorization of requirements. The categories are colored for user interface purpose. Note that a requirement may be assigned to several of these categories. The principle use of this attribute is to allow filtering within the various views, in order to manage large sets of requirements efficiently.
IE Risk Impact	enum multival	Operational Use Performance Cost Timescale Technology Organization Delivery	This attribute is used to classify the type of risk, if any, associated with a requirement. Note that a requirement may be assigned to several of these categories.
IE Risk Consequence	enum	Negligible Marginal Significant Critical Catastrophic	This attribute is used to assign an impact level to the risk.

Attribute name	Type	value	Description
IE Risk Probability	enum	Negligible Low Likely very Likely Inevitable	This attribute is used to assign a probability of occurrence to a risk.
IE Key Requirement	enum	true false	This attribute is used to identify the given Requirement as a key Requirement. Alternatively, a project could delete this attribute and add “Key” to the “IE Req Priority” enumerations in the “IE Level” type.
IE Phase	string	see description	This attribute is used to define the increment phase in which the requirement is implemented into a system/product. Typically it is used to define the future release at which new functionality will be available. It may be appropriate for some projects to control this attribute more precisely by setting up an enumerated list of approved future releases.
IE Req Status	enum	In negotiation Accepted Analysis Obsolete	This attribute is used to define the progress status of the requirement analysis activity..
IE Object Version	string	see description	This attribute is managed by DXL, specifically by the “Update Version Attribute ” utility. The attribute holds the designation of the baseline of the module in which the object was last changed.
IE Rationale	Text		This attribute is used to record "routine" decisions and issues within the responsibility of individual engineers or co-located teams. (Major issues and decisions should be recorded in the associated issue/Decision module)

Available relationships for User Requirements objects

link	link way	target type	link module	Description
Compliance	Incoming	Requirement	satisfies	The linked requirements satisfy the given User Requirement.
Issue raised	Incoming	Issue-Decision	refers to	Shows that this User Requirement is to be (or has been) analyzed through a Issue and Decision process.
Verification	Incoming	IVV Procedure	verifies	The linked IVV Procedures verify the given User Requirement.

Available views of User Requirements module

View name	filter/sort available	Description
Standard view	none	DOORS default view, which shows the DOORS ID and Object Heading/Text.
Associated issues	inactive filter, no sort: IE Object Type == Requirement	Displays the attributes: PUID, Object Text, Compliance, Status, Flexibility, Priority, Risk Impact, Rationale, and also a traceability column obtained with the objects reached by an incoming "refers to" link. This traceability column is headed “referred by…” and (for each link found) shows the name of the module referred to, the PUID of the object referred to, and its Object Text.
Compliance matrix	active filter, no sort: IE Object Type == Requirement	Displays the attributes: PUID, Object Text, Category, Compliance, Priority, Phase, Risk Impact, and also a traceability column obtained with the objects reached by an incoming "satisfies" link. This traceability column is headed “satisfied by…” and (for each link found) shows the name of the module referred to, the PUID of the object referred to, and its Object Text.
Risk analysis	active filter and active sort: IE Object Type == Requirement and IE Risk Impact != NULL sorting by IE Risk Consequence	Displays the attributes: Risk Level (as a LED chart), PUID, Object Text, Category, Status, Risk Impact, Risk Level and probability of occurrence.
Document view	inactive filter, no sort: IE Object Type == Requirement	Displays the attributes: Document Style, PUID and Object Heading/Text.
Key requirements list	active filter, no sort: IE Object Type == Requirement and IE Key Requirement == true	Displays the attributes: PUID, Key Requirement, Object Text, Category, Compliance, Status, Flexibility, Priority, Phase and Risk Impact.
Requirements analysis	inactive filter, no sort: IE Object Type == Requirement	Displays the attributes: PUID, Object Text, Category, Compliance, Status, Flexibility, Priority, Key Requirement, Phase, Risk Impact and Rationale.
Requirements in negotiation	active filter, no sort: IE Object Type == Requirement and IE Req Status == in negotiation	Displays attributes PUID, Object Text, Category, Compliance, Status, Flexibility, Priority, Phase and Risk Impact.
Validation matrix	active filter, no sort: IE Object Type == Requirement	Displays the attributes: PUID, Object Text, Flexibility, Compliance, Status, Priority, Risk Impact and Risk Level, and also 2 traceability columns obtained with the IVV Procedures reached by an incoming "verifies" link, one for the IVV Procedures description and one for the Verification Method. The first traceability column is headed “verified by…” and (for each link found) shows the name of the module referred to, the PUID of the object referred to, and its Object Text. The second traceability column is headed “Verification Method” and just shows the code I, A, T, D or null as appropriate.

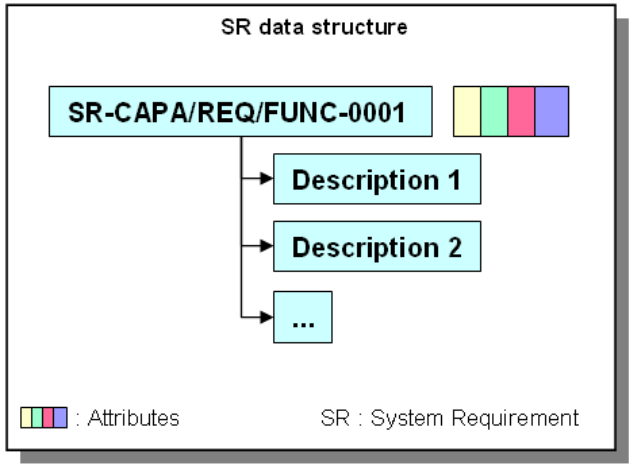
Appendix B. System Requirements Module Description Form

This module description form contains information about DOORS formal module of type "System Requirements". SR Module is used to describe System specification document. It could also be used for Subsystem or Prime Item specification documents.

The "System Requirements" Module contains RMF Objects of types “Capability”, "Requirement", and/or “Function”. The data structure for each such object is a tree. The root DOORS object is the RMF Object itself. It carries all the attributes as it is the only DOORS object in the tree, which can be identified as a RMF object and gain a PUID. All links to and from the object should be attached to the root object

The leaves of the tree are DOORS objects which inherit the attribute "IE Object Type", may have the value “Capability”, "Requirement", or “Function”. The leaf DOORS objects should only be used for descriptive or explanatory material, and all of their RMF attributes should be set to ‘blank’. Leaf objects cannot be assigned a PUID, and should not be linked using standard RMF linksets.

The enumerated lists shown in the following tables are the defaults supplied with the delivered IRDRMFAO. They should all be customised to be appropriate to the individual project.



Typical setup/process for the "System Requirements" Module

1) Assumption : Originating Document is available from WORD. It could be any other format readable by DOORS (see import format supported).

2) Create a new empty "User Requirements" module, by restoring (using file>restore>module in the Database Window) the file SR.dma.

3) From WORD, export the document to DOORS, by clicking the "Export To Doors" icon in the WORD tool bar,

4) Within DOORS, select paragraphs that represent capabilities, requirements or functions and use the IRDRMFAO interface "Manage Objects” to assign the appropriate object type.

5) For each RMF object, fill in the attributes using the DOORS views..

Module attributes of System Requirements module

Attribute name	Type	value	Description
IE Mod Type	enum	System Requirements Subsystem Requirements Prime Item Requirements	Type of the module : "System Requirements", "Subsystem Requirements" or "Prime Item Requirements". The Value is set by default to "System Requirements" but could be changed on need. For the whole list of values, see the "Project Profile" Module Description.
IE Mod LifeCycle Phase	enum	Feasibility Study Model Simulation Demonstrator Prototype Full Development Production Operation & Support	Represents the life cycle phase of the project to which the objects identified in the module apply.
IE Capability Number	Integer		An internal counter used by IRDRMFAO, do not edit.
IE Requirement Number	Integer		An internal counter used by IRDRMFAO, do not edit.
IE Function Number	Integer		An internal counter used by IRDRMFAO, do not edit.
IE Release	string	see description	Stores the IRDRMFAOrelease which has been used to create this module.

Object attributes of System Requirements module

These attributes apply mostly to the requirements (i.e. "IE Object Type" attribute is "Requirement"). It may not be appropriate to apply all of them to capabilities or functions (i.e. "IE Object Type" attribute is "Capability" or “Function”) .

Attribute name	Type	value	Description
IE Object Type	enum	Requirement Function Capability	Type of the object, null if object is not identified as a RMF root object.. For the whole list of values, see the "Project Profile" Module Description.
IE PUID	string	see description	Project Unique ID. This attribute is automatically set with the following concatenation rule : [module prefix]-[object prefix]-[number] - the [module prefix] is obtained from the DOORS module attribute "Module Prefix", - the [object prefix] is obtained from the DOORS attribute "IE Object Prefix" (see the "Project Profile" Module Description), - the [number] is incremented by one every new RMF object.
IE Req Priority	enum	Low Medium High	This attribute represents the level of priority the System Engineer has set to the requirement.
IE Req Compliance	enum	Not Met Partially Met Met	This attribute represents the level of compliance obtained from the succeeding level in the design hierarchy. If compliance is not “Met”, there should be a commentary in the “Rationale” attribute or a specific issue raised in the appropriate Issue-Decision module.
IE Req Flexibility	enum	Low Medium High	Represent the level of flexibility the system engineer could have when negotiating the requirement with a sub-contractor.

Attribute name	Type	value	Description
IE Req Category	enum multival	Mission Operational Objective Operational Scenario	Allow categorization of capabilities, requirements and functions. The categories are colored for user interface purpose.

		Functional - Function - Data - Behavior Performance Dependability - Reliability - Availability - Maintainability - Security - Safety Constraints - Cost - Schedule - Architecture - IV&V - Development - Production - Delivery - Ergonomy - Installation	
IE Risk Impact	enum multival	Operational Use Performance Cost Timescale Technology Organization Delivery	This attribute is used to classify the type of risk, if any, associated with a capibility, requirement or function. Note that several classifications amy be assigned a single RMF object.
IE Risk Level	enum	Negligible Marginal Significant Critical Catastrophic	This objecxt is used to assign an impact level to the risk.

Attribute name	Type	value	Description
IE Risk Probability	enum	Negligible Low Likely Very Likely Inevitable	This attribute is used to assign a probability of occurrence to a risk.
IE Key Requirement	enum	true false	This attribute is used to identify the given Capability, Requirement or Function as a Key requirement. This usually means that the requirement has a strong influence on cost, schedule, functionality, risk or performance.
IE Phase	string	see description	This attribute is used to define the increment phase in which the requirement is implemented into a system/product. Typically it is used to define the future release at which new functionality will be available. It may be appropriate for some projects to control this attribute more precisely by setting up an enumerated list of approved future releases.
IE Req Tolerance	string	see description	This attribute defines the tolerance for the requirement (performance requirements only).
IE Req Status	enum	In negotiation Accepted Analysis Obsolete	This attribute is used to define the progress of the requirement analysis and/or system design activity.
IE Req Type	enum	Originating Deriv:ed Induced Calculated	This attribute defines the type of the requirement, it is useful to record how the requirement was created. The followingdefinitions, derived from SYS-EM, are offered as IRDRMFAO standards: Originating: Can be traced with little or no modificastion to a higher level requirement. Derived The result of functional partitioning of a higher level requirement. Induced: A new requirement produced as a result of a design decision at this level. Calculated:. The result of numerical partitioning of a higher level performance parameter.
IE Object Version	string	see description	This attribute is managed by DXL, specifically by the “Update Version Attribute” utility. The attribute holds the designation of the baseline of the module in which the object was last changed.
IE Rationale	Text	See description	This attribute records "routine" decisions and issues within the responsibility of individual engineers or co-located teams.

Available relationships for System Requirements objects

Link	link way	target type	link module	Description
Allocation	incoming	Configuration Item	is allocated by	the given capability, requirement or function is allocated to the linked configuration items.
Compliance	outgoing	Requirement	satisfies	the linked upper requirements are satisfied by the given requirement or function.
Justification	outgoing	Issue-Decision	is justified by	represent the justification of the Capability/Requirement/function object, i.e. link to a decision.
Issue raised	incoming	Issue-Decision	refers to	Shows that the Capability/Requirement/Function is to be (or has been) analyzed through an Issue and Decision process.
Verification	incoming	IVV Procedure	verifies	Shows which IVV procedures verify the given Capability/Requirement/Function.
Composition	internal	Capability/ Requirement/ Function	Is composed of	An internal link to the SR module which is used to establish a Capability > Requirement > Function hierarchy. Each link should be sourced at the higher level object (which is contrary to the convention established for external links).

Available views of System Requirements module

View name	filter/sort available	Description

Standard view	none	The DOORS default view, which shows the DOORS ID and Object Heading/Text.
Allocation matrix	active filter, no sort: IE Object Type != NULL	Displays the attributes: PUID, Object Text, Category, Compliance, Status, Priority, Phase, Risk Impact, and also a traceability column obtained with the objects reached by an incoming "is allocated by" link. This traceability column is headed “allocated to…” and (for each link found) shows the name of the module referred to, the PUID of the object referred to, and its Object Text.
Associated issues	inactive filter, no sort: IE Object Type != NULL	Displays the attributes: PUID, Object Text, Compliance, Type, Status, Flexibility, Priority, Risk Impact, and also a traceability column obtained with the objects reached by an incoming "refers to" link. This traceability column is headed “referred by…” and (for each link found) shows the name of the module referred to, the PUID of the object referred to, and its Object Text.
Compliance matrix	active filter, no sort: IE Object Type == Requirement	Displays the attributes: PUID, Object Text, Category, Compliance, Status, Priority, Phase, Risk Impact, and also a traceability column obtained with the objects reached by an incoming "satisfies" link. This traceability column is headed “satisfied by…” and (for each link found) shows the name of the module referred to, the PUID of the object referred to, and its Object Text.
Risk analysis	active filter and active sort: IE Object Type == Requirement and IE Risk Impact != NULL sorting by IE Risk Level	Displays the attributes: Risk Level (as a LED chart), PUID, Object Text, Status, Risk Impact, Risk Level and probability of occurence.
Document view	inactive filter, no sort: IE Object Type != NULL	Displays the attributes: Document Style, PUID and Object Text.
Justification	inactive filter, no sort: IE Object Type != NULL	Displays the attributes PUID, Object Text, rationale, Compliance, Type, Status, Flexibility, Priority, Risk Impact, and also a traceability column obtained with the objects reached by a "is justified by" outgoing link. .This traceability column is headed “is justified by…” and (for each link found) shows the name of the module referred to, the PUID of the object referred to, and its Object Text.
Key requirements list	active filter, no sort: IE Object Type == Requirement and IE Key Requirement == true	Displays the attributes: PUID, Key Requirement, Object Text, Category, Compliance, Type, Status, Flexibility, Tolerance, Priority, Phase and Risk Impact.
Requirements analysis	inactive filter, no sort: IE Object Type != NULL	Displays the attributes: PUID, Object Text, Category, Compliance, Type, Status, Flexibility, Tolerance, Priority, Phase, Key Requirement, Risk Impact and Rationale.
Requirements in negotiation	active filter, no sort: IE Object Type == Requirement and IE Req Status == in negotiation	Displays the attributes: PUID, Object Text, Category, Compliance, Status, Flexibility, Priority, Phase and Risk Impact.
Upper requirements satisfied	active filter, no sort: IE Object Type == Requirement	Displays the attributes: PUID, Object Text, Category, Type, Status, Phase and Risk Impact, and also a traceability column obtained with the objects reached by a "satisfies" outgoing link. This traceability column is headed “satisfies…” and (for each link found) shows the name of the module referred to, the PUID of the object referred to, and its Object Text.
Verification matrix	active filter, no sort: IE Object Type == Requirement	Displays the attributes: PUID, Object Text, Flexibility, Compliance, Status, Priority, Risk Impact and Risk Level, and also 2 traceability columns obtained with the IVV Procedures reached by an incoming "verifies" link, one for the IVV Procedures description and one for the Verification Method. The first traceability column is headed “verified by…” and (for each link found) shows the name of the module referred to, the PUID of the object referred to, and its Object Text. The second traceability column is headed “Verification Method” and just shows the code I, A, T, D or null as appropriate

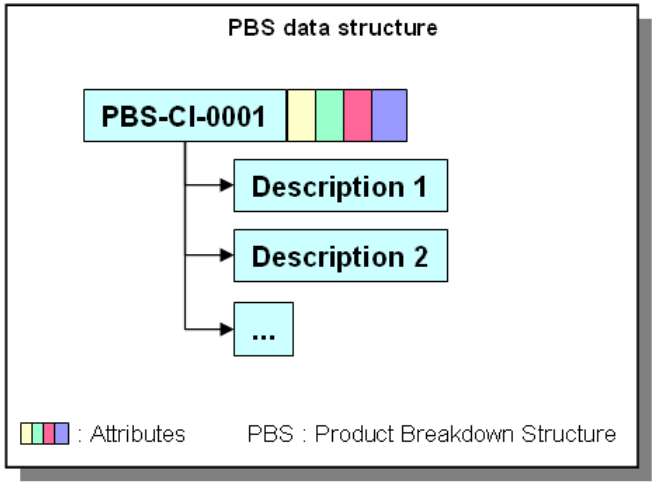
Appendix C. PBS Module Description Form

This module description form contains information about DOORS formal module of type "Product Breakdown Structure" (PBS).

The "PBS" Module contains RMF Objects of type “Configuration Item”. The data structure for each such object is a tree. The root DOORS object is the RMF Object itself. It carries all the attributes as it is the only DOORS object in the tree, which can be identified as a RMF object and gain a PUID. All links to and from the object should be attached to the root object

The leaves of the tree are DOORS objects which inherit the attribute "IE Object Type", may have the value “Configuration Item”. The leaf DOORS objects should only be used for descriptive or explanatory material, and all of their RMF attributes should be set to ‘blank’. Leaf objects cannot be assigned a PUID, and should not be linked using standard RMF linksets.

The enumerated lists shown in the following tables are the defaults supplied with the delivered IRDRMFAO. They should all be customised to be appropriate to the individual project



Typical setup/process for the "Product Breakdown Structure" Module

1) Assumption : Originating Document is available from WORD. It could be any other format readable by DOORS (see import format supported).

2) Create a new empty "PBS" module, by restoring (using file>restore>module in the Database Window) the file PBS.dma.

3) From WORD, export the document to DOORS, by clicking the "Export To Doors" icon in the WORD tool bar,

4) Within DOORS, select paragraphs that configyration items and use the IRDRMFAO interface "Manage Objects” to assign the appropriate object type.

For each RMF object, fill in the attributes using the DOORS views..

Module attributes of PBS module

Attribute name	Type	value	Description
IE Mod Type	enum	Product Breakdown Structure	Type of the module, always "Product Breakdown Structure" in this case. For the whole list of values, see the "Project Profile" Module Description.
IE Mod LifeCycle Phase	enum	Feasibility Study Model Simulation Demonstrator Prototype Full Development Production Operation & Support	Represents the life cycle phase of the project to which the objects identified in the module apply.
IE Configuration Item Number	Integer		An internal counter used by IRDRMFAO, do not edit.
IE Release	string	see description	Stores the IRDRMFAO release which has been used to create this module.

Object attributes of PBS module

These attributes apply to the "Configuration Item" objects (i.e. "IE Object Type" attribute is "Configuration Item")

Attribute name	Type	value	Description
IE Object Type	enum	Configuration Item	Type of the object, always "Configuration Item" in this case, or null. For the whole list of values, see the "Project Profile" Module Description.
IE PUID	string	see description	Project Unique IDentifier. This attribute is automatically set with the following concatenation rule : [module prefix]-[CI prefix]-[number] - the [module prefix] is obtained from the DOORS module attribute "Module Prefix", - the [CI prefix] is obtained from the DOORS attribute "IE Object Prefix" (see the "Project Profile" Module Description), - the [number] is incremented by one every new RMF Object.
IE CI Type	enum	System Subsystem Prime Item HWCI CSCI Interface NDI COTS	Represent the type of the Configuration Item. This list shows sample items, each project should define an appropriate list. A ‘Platform’ CI type is often useful.
IE CI Serial Number	string	see description	Represent the identification of the Configuration Item as identified by the Configuration Management function. The numbering will be project specific.
IE CI Version Number	string	see description	Represent the version of the Configuration Item as identified by the Configuration Management function. The numbering will be project specific.
IE CI Revision Number	string	see description	Represent the revision of the Configuration Item as identified by the Configuration Management function. The numbering will be project specific.
IE CI Supplier Id	string	see description	Represent the identification of the Supplier of the Configuration Item as identified by the Configuration Management function.
IE Object Version	string	see description	This attribute is managed by DXL, specifically by the “Update Version Attribute ” utility. The attribute holds the designation of the baseline of the module in which the object was last changed.

Available relationships for PBS objects

link	link way	target type	link module	Description

allocation	outgoing	Capability , Requirement,or Function	is allocated by	The linked capabilities, requirements or functions are allocated to the given configuration item.
justification	outgoing	Issue-Decision	is justified by	Represents the justification of the Configuration Item object, i.e. link to a decision.
Issueraised	incoming	Issue-Decision	refers to	Shows that the Configuration Item is to be (or has been) analyzed through a Issue and Decision process.
Design hierarchy	None	Configuration Item	None	The DOORS Outline heading numbering can be used to establish a design hierarchy for a system.

Available views of PBS module

View name	Filter	Description
Standard view	None	The DOORS default view, which shows the DOORS ID and Object Heading/Text.
Allocated requirements	active filter, no sort: IE Object Type == Configuration Item	Displays the attributes: PUID, Object Text, Type, and also a traceability column obtained with the objects reached by a "is allocated by" outgoing link. This traceability column is headed “allocated by…” and (for each link found) shows the name of the module referred to, the PUID of the object referred to, and its Object Text.
Associated issues	inactive filter, no sort: IE Object Type == Configuration Item	Displays the attributes: PUID, Object Text, Type, and also a traceability column obtained with the objects reached by an incoming "refers to" link. This traceability column is headed “referred by…” and (for each link found) shows the name of the module referred to, the PUID of the object referred to, and its Object Text.
Configuration Item Analysis	active filter, no sort : IE Object Type == Configuration Item	Displays the attributes: PUID, Object Text, Type, Supplier, CM identification, CM version and CM revision.
Document view	inactive filter, no sort : IE Object Type == Configuration Item	Displays the attributes: Document Style, PUID and Object Text.
Justification	inactive filter, no sort: IE Object Type == Configuration Item	Displays the attributes: PUID, Object Text, Type, and also a traceability column obtained with the objects reached by a "is justified by" outgoing link. This traceability column is headed “is justified by…” and (for each link found) shows the name of the module referred to, the PUID of the object referred to, and its Object Text.

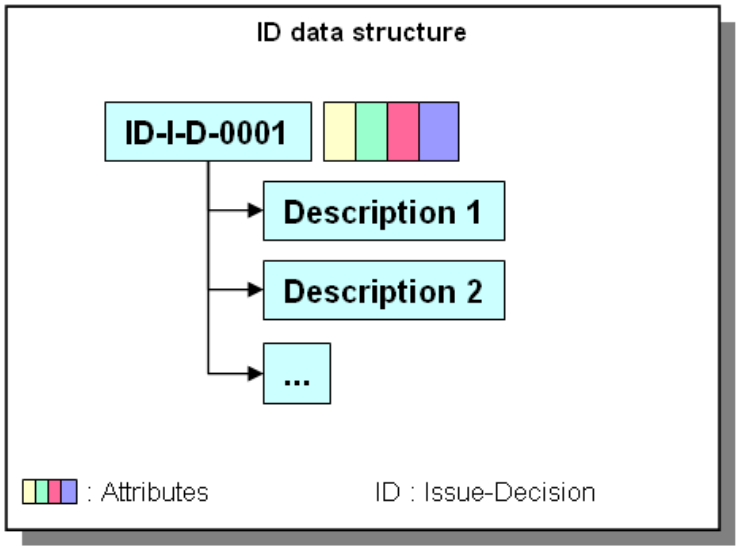
Appendix D. Issues and Decisions and Justifications Module Description Form

This module description form contains information about DOORS formal module of type "Issues and Decisions" and Justifications. This type of module is used for different analysis purposes like Requirements Analysis, Design Analysis, etc.

The "IDJ" Module contains RMF Objects of type “Issue-Decision”. The data structure for each such object is a tree. The root DOORS object is the RMF Object itself. It carries all the attributes as it is the only DOORS object in the tree, which can be identified as a RMF object and gain a PUID. All links to and from the object should be attached to the root object

The leaves of the tree are DOORS objects which inherit the attribute "IE Object Type", may have the value “Issue-Decision”. The leaf DOORS objects should only be used for descriptive or explanatory material, and all of their RMF attributes should be set to ‘blank’. Leaf objects cannot be assigned a PUID, and should not be linked using standard RMF linksets.

The enumerated lists shown in the following tables are the defaults supplied with the delivered IRDRMFAO. They should all be customised to be appropriate to the individual project



Typical setup/process for the "Issue-Decision" Module

1) Assumption : Originating Document is available from WORD.
2) Create a new empty "IDJ" module, by restoring the file IDJ.dma.
3) From WORD, export the document to DOORS, by clicking on the "Export To Doors" icon,
4) Within DOORS, use the IRDRMFAO interface "Manage Objects” to assign appropriate type.

Module attributes of Issues and Decisions module

Attribute name	Type	value	Description
IE Mod Type	enum	Issues and Decisions Requirement Analysis ID Design Analysis ID Operational Concepts	For the whole list of values, see the "Project Profile" Module Description.
IE Mod LifeCycle Phase	enum	Feasibility Study Model Simulation Demonstrator Prototype Full Development Production Operation & Support	Represents the life cycle phase of the project to which the objects identified in the module apply.
IE Issue-Decision Number	Integer		An internal counter used by IRDRMFAO , do not edit.
IE Release	string	see description	Stores the IRDRMFAO release which has been used to create this module.

Object attributes of Issues and Decisions module

These attributes apply to the "Issue and Decision" objects (i.e. "IE Object Type" attribute is "Issue-Decision")

Attribute name	Type	value	Description
IE Object Type	enum	Issue-Decision Justification	Type of the object, always "Issue-Decision" in this case, or null. For the whole list of values, see the "Project Profile" Module Description.
IE PUID	string	see description	Project Unique IDentifier. This attribute is automatically set with the following concatenation rule : [module prefix]-[I-D prefix]-[number] - the [module prefix] is obtained from the DOORS module attribute "Module Prefix", - the [I-D prefix] is obtained from the DOORS attribute "IE Object Prefix" (see the "Project Profile" Module Description), - the [number] is incremented by one every new Issue-Decision RMF object.
IE I-D Priority	enum	Low Medium High	This attribute represents the level of priority set to the Issue-Decision.
IE I-D Decision	text	see description	This attribute records the decision associated with the Issue-Decision.
IE I-D Decision date	date	see description	This attribute records the date of the decision associated with the Issue-Decision.
IE I-D Status	enum	In negotiation Accepted Analysis Obsolete	This attribute is used to define the progress of the Issue-Decision.
IE Object Version	string	see description	This attribute is managed by DXL, specifically by the “Update Version Attribute ” utility. The attribute holds the designation of the baseline of the module in which the object was last changed.

Available relationships for Issues and Decisions objects

link	link way	target type	link module	Description
justification	incoming	All types of object, except Issue-	is justified by	This relationship represents the justification for an object, i.e. a decision. The Issue-Decision object justifies the linked source objects.

		Decision objects		
Issue raised	outgoing	All types of object, except Issue-Decision objects .	refers to	This relationship represents the fact that he linked target objects are (or have been) the subject of an issue-decision process.

Available views of Issues and Decisions module

View name	filter	Description
Standard view	none	The DOORS default view, which shows the DOORS ID and Object Heading/Text..
Decisions justify...	active filter, no sort : IE Object Type == Issue-Decision	Displays the attributes: PUID, Object Text, Status, and also a traceability column obtained with the objects reached by a "is justified by" incoming link. This traceability column is headed “decision justifies…” and (for each link found) shows the name of the module referred to, the PUID of the object referred to, and its Object Text.
Document view	inactive filter, no sort : IE Object Type == Issue-Decision	Displays the attributes: Document Style, PUID and Object Text.
Issues and Decisions Analysis	inactive filter, no sort : IE Object Type == Issue-Decision	Displays the attributes: PUID, Object Text, Decision, Priority, Status and Decision Date
Issues refer to...	active filter, no sort : IE Object Type == Issue-Decision	Displays the attributes: PUID, Object Text, Status, and also a traceability column obtained with the objects reached by a "refers to" outgoing link. This traceability column is headed “Issue refers to…” and (for each link found) shows the name of the module referred to, the PUID of the object referred to, and its Object Text.

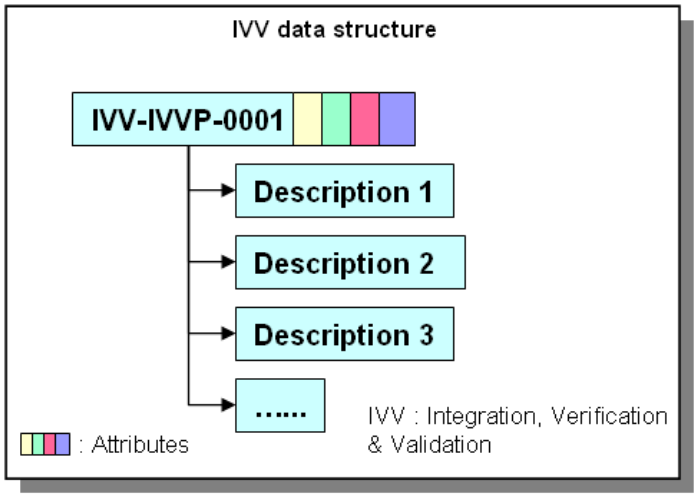
Appendix E. IVV Procedures Module Description Form

This module description form contains information about the DOORS formal module of type "Integration, Verification or Validation Procedures", as delivered with IRDRMFAO.

The "IVV Procedures" Module contains RMF objects of type "IVV Procedure". The data structure for each such object is a tree. The root DOORS object is the IVV Procedure itself. It carries all the attributes applying to the IVV Procedure, it is the only DOORS object in the tree, which can be identified as a RMF object and gain a PUID. All links to and from the object should be attached to the root object

The leaves of the tree are DOORS objects which inherit the attribute "IE Object Type", which for an IVV Procedure, has the value "IVV Procedure". The leaf DOORS objects should only be used for descriptive or explanatory material, and all of their RMF attributes should be set to 'blank'. Leaf objects cannot be assigned a PUID, and should not be linked using standard RMF linksets.

The standard IRDRMFAO 'as delivered' module attributes, object attributes, attribute types, link relationships and views are defined in the following tables. These should all be tailored to the particular project when it is set up..



Suggestion: - The basic RMF IVV Procedure is designed to specify a group of tests which can be planned and approved as a whole. At a later stage individual tests will need to be specified in sufficient detail that test engineers can execute them and the results can be validated. The operating instructions and expected results for the detailed tests could be entered as leaves within the RMF IVV object, and a test specification document created using the Enhanced Word Exporter utility.

Module attributes of IVV Procedures template module

Attribute name	Type	Value	Description
IE Mod Type	Enum	Integration Procedures Verification Procedures Validation Procedures	Type of the module, based on the IVV template. Additional types could be added, but these must be first defined as Module Types in the "Project Profile" Module Description.
IE Mod LifeCycle Phase	Enum	Feasibility Study Model Simulation Demonstrator Prototype Full Development Production Operation & Support	Represents the life cycle phase of the project applying to the verification procedures identified in the module.
IE Release	String	See description	Stores the IRDRMFAO release which has been used for create this module.
IE IVV Procedure Number	Integer		An internal counter used by IRDRMFAO, do not edit.
Prefix	String	IVV	Module prefix, used by IRDRMFAO to generate the PUID. In IRDRMFAO v2.0, this must be edited manually if necessary. In later versions it may be driven from the Module Object in the "Project Profile" Module Description

Object attributes of IVV Procedures module

These attributes apply to the IVV Procedures (i.e. "IE Object Type" attribute is "IVV Procedure")

Attribute name	Type	Value	Description
IE Object Type	Enum	IVV Procedure	Type of the object, always "IVV Procedure" in the delivered IRDRMFAO, or null. Additional types could be added, but these must first be defined as Object types in the "Project Profile" Module Description.
IE PUID	String	See description	Project Unique ID. This attribute is automatically set with the following concatenation rule: [module prefix]-[IVV prefix]-[number] - the [module prefix] is obtained from the DOORS module attribute "Module Prefix". - The [IVV prefix] is obtained from the DOORS object attribute "IE Object Prefix" (see the "Project Profile" Module Description). - The [number] is incremented by one every new verification procedure object created or identified.
IE IVV Type	Enum	Prototype production prototype and production	For each IVV Procedure, this attribute records the kind of the procedure. Production tests are those intended to prove that each product has been built correctly to its design, prototype tests are intended to prove that the design meets the product specifications.
IE IVV Approval Level	Enum	Customer Prime Contractor Sub-Contractor Independent Test House	For each IVV Procedure, this attribute defines which party to the contract is to be responsible for the approval of the test specifications and the test results.
IE IVV Responsible	String		Use this attribute to store the identification of the person responsible in charge of the given IVV Procedure (from the engineering or test organizations).
IE IVV Skills	Text		Represents the list of specific skills needed for the IVV procedure, or identification of the people having these specific skills (for instance, special operation or equipment skills).
IE IVV Event	String		Defines the event or location at which the procedure will be executed. Examples are ‘supplier factory test’, ‘customer platform test’, ‘prime contractor integration facility’. Suggestion: - Projects could set this attribute up as an enumerated type.
IE IVV Event Provider	Enum	customer	Represents the party responsible for the conduct of the IVV event.).

		prime contractor sub contractor	Suggestion: - Projects could set this attribute up as an enumerated type, giving named sub-contractors etc.
IE IVV Non Regression	Enum	true false	Allow identification of non-regression IVV procedures. These are procedures that are unlikely to have to be repeated in the event of a change to the system.
IE IVV Method	Enum	I A D T	Identify the verification method of the requirements (**I, A, D, T methods) :** **Inspection** (for example, inspection of drawings), **Analysis** (for example, using simulation or virtual reality prototype), **Demonstration** (for example, using mock-ups or physical models), or **Test** (for example, by testing physical prototypes, breadboards)
IE IVV Scope	Text		This attribute is used to record agreement with customers and suppliers on the general scope and objectives of the IVV activity before too much effort is expended on the creation of detail test specifications.
IE IVV Pre and Post test Actions	Text		This attribute is used define and constrain significant actions necessary before and after the execution of the IVV procedure. Examples are commitment to deliver specifications for approval at a minimum time ahead of the event, and similarly commitments to approve documents in specific timescales.
IE IVV Acceptance criteria	Text		Used to define general principles of acceptance criteria for the group of tests within this procedure, for example 'what statistical confidence level needs to be achieved', 'what allowance for instrument error is acceptable?
IE Object Version	String	See description	This attribute is managed by DXL, specifically by the "Update Version Attribute " utility. The attribute holds the designation of the baseline of the module in which the object was last changed.

Available relationships for IVV Procedures objects

Link	Link way	target type	Link module	Description
Justification	Outgoing	Issue-Decision	is justified by	Represents the justification of the IVV Procedure objects i.e. link to a decision.
under issue process	Incoming	Issue-Decision	Refers to	Shows that this IVV Procedure is to be (or has been) analyzed through an Issue and decision process.
Verification	Outgoing	Requirement	Verifies	Shows which requirements are verified by this IVV procedure.
Allocation	Incoming	PBS (test equip)	Is allocated by	Shows which test or other support equipment is allocated to this procedure.

Available views of IVV Procedures module

View name	Filter/sort available	Description
Standard view	None	DOORS default view, which shows the DOORS ID and Object Heading/Text
Associated issues	Inactive filter, no sort: IE Object Type == IVV Procedure	Display attributes PUID, Object Text, Method, Type, and also a traceability column obtained with the objects reached by an incoming "refers to" link. This traceability column is headed "referred by…" and (for each link found) shows the name of the module referred to, the PUID of the object referred to, and its Object Text
Document View	Inactive filter, no sort: IE Object Type == IVV Procedure	Displays the attributes: Document Styles, PUID and Object Header/Text.
IVV Matrix View	Active filter, no sort: IE Object Type == IVV Procedure	Displays the attributes: PUID, Object Text, Method, Type, and also a traceability column obtained with the Requirements reached by an incoming "verifies" link. . This traceability column is headed "verifies…" and (for each link found) shows the name of the module referred to, the PUID of the object referred to, and its Object Text
IVV Procedure Definition View	Inactive filter, no sort: IE Object Type == IVV Procedure	Displays the attributes: PUID, Object Text, Scope/Objectives, Method, Type, Approval, Acceptance Criteria, Pre & Post Event Actions.

IVV Procedure Planning View	Inactive filter, no sort: IE Object Type == IVV Procedure	Displays the attributes: PUID, Object Text, Responsible, Event, Event provider, Specific skills needed and Non-Regression.
Justification View	Inactive filter, no sort: IE Object Type == IVV Procedure	Displays the attributes: PUID, Object Text, Method, Type, and also a traceability column obtained with the objects reached by a "is justified by" outgoing link. This traceability column is headed “is justified by…” and (for each link found) shows the name of the module referred to, the PUID of the object referred to, and its Object Text
Test Equipment View	Inactive filter, no sort: IE Object Type == IVV Procedure	Displays the attributes: PUID, Object Text, Method, Type and also a traceability column obtained with the objects reached by a "is allocated by" incoming link. This traceability column is headed “Support items needed” and (for each link found) shows the name of the module referred to, the PUID of the object referred to, and its Object Text

Appendix F. Project Profile Module Description Form

This module description form contains information about DOORS formal module named "Project Profile".

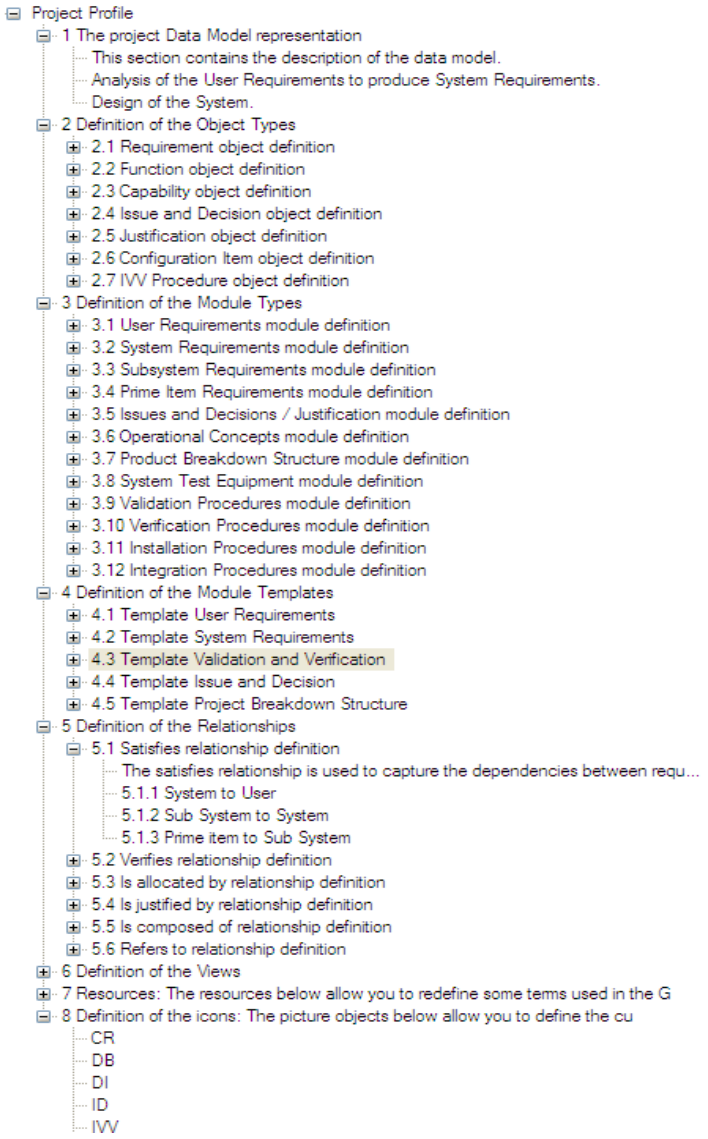
The "Project Profile" module defines the available object types, module types and relationships usable for the current RMF project, together with other project standards such as paragraph styles. In general it controls the DOORS project by defines the drop down lists presented by the IRDRMFAO utilities. The objects represent all the data suitable for System Engineering, like requirements, capabilities, critical issues, etc. The modules contain the objects. Typical modules are User Requirements, System Requirements, Issue-and Decision, etc.

Typical setup/process for creating a new project

1) To create a new RMF project, run DOORS and from the DOORS Database window, run the IRDRMFAO utility “Create a new RMF project”. Type the project name and description when prompted by the utility.

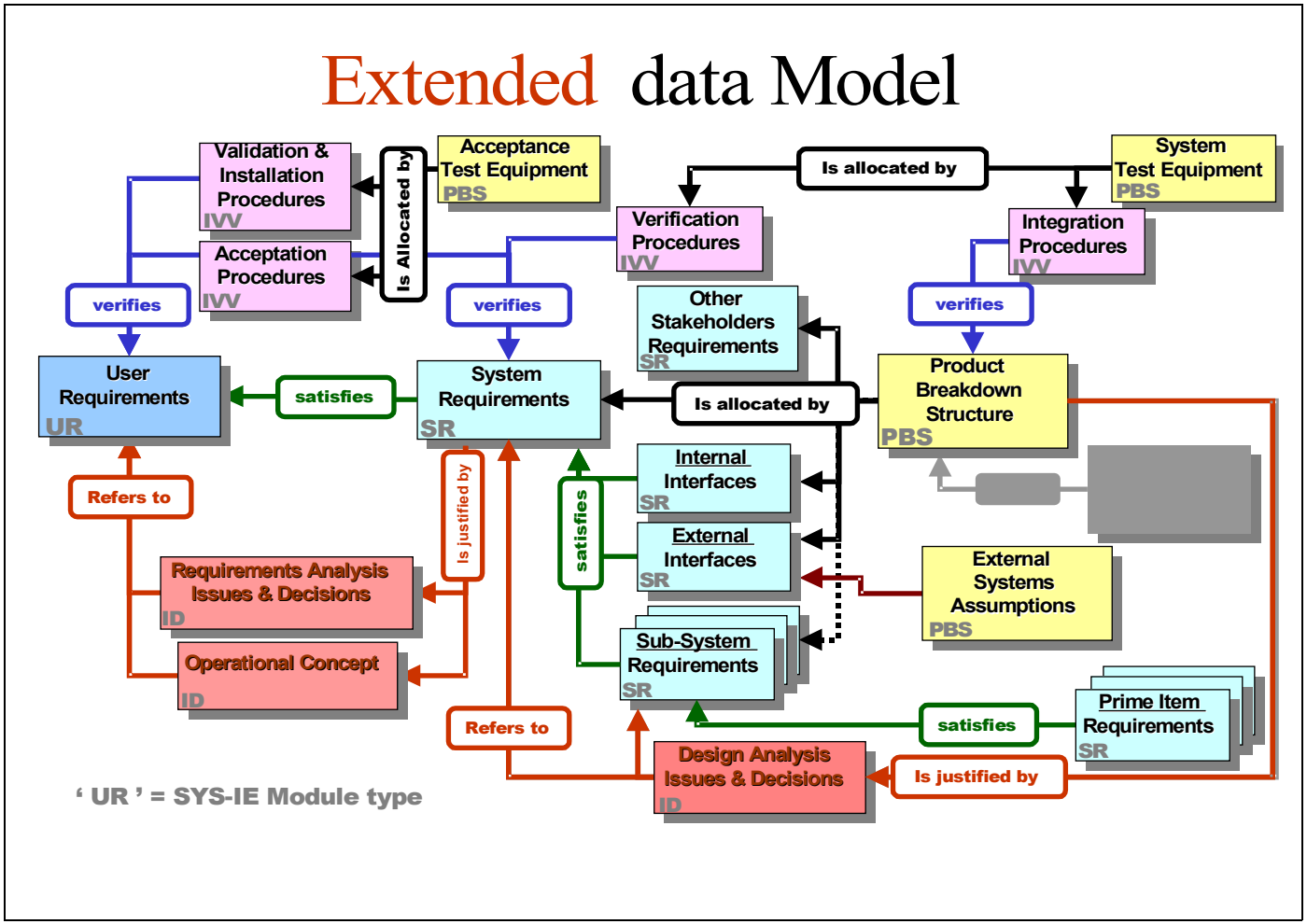
2) Your project contains now a formal module named "Project Profile" and-eight link modules "is allocated by", "is composed of", “is different to”, “is identical to”, "is justified by", ,refers to",."satisfies", and "verifies".

Read the descriptive paragraphs in the Project Profile module which define the default customization provided with IRDRMFAO, as delivered. If necessary, change the values of attribute types, add new object types or new module types in order to further customize the model for your project. Do not forget to update the data model diagram as necessary.



Definition of the Data Model

One Data Model currently supported by this IRDRMFAO version is shown below. The diagram provided within the delivery of IRDRMFAO shows a slightly simpler model. Projects should produce a specific data model which represents how System Engineering is to be carried out It is anticipated that the project data model should be defined, documented and approved as part of the Systems Engineering Management Plan (SEMP)



Module attributes of Project Profile module

These attributes apply to the module, and are interpreted to apply throughout the project by IRDRMFAO utilities.

Attribute name	Type	value	Description
IE Log File	string	See description default is errlog.txt	Pathname of log file (full or not, e.g. C:\doors\errlog.txt or errlog.txt), which can be set by the user if desired. The default path is DOORSHOME errlog.txt which is always used if the Project Profile module is missing or the Log File attribute is missing from it.
IE Debug	enum	No Debug DXL Window Log File Ack and Log Window and Log	Flag indicating where error messages should be directed. In No Debug mode, tools ack and halt. Otherwise they attempt to continue.
IE Data Model	text		This attribute is not used any more

Object attributes of Project Profile module

The following attributes apply to the definition of objects, and are set to null in other objects.

Attribute name	Type	value	Description
IE Object Type	string enum	see description. Typical values are Requirement, Capability, Issue-Decision, Configuration Item, IVV Procedure.	Represents an object type usable in the current project. The objects with this attribute set to a value (i.e. not NULL) describe the data definition of objects suitable for System Engineering, like requirements, capabilities, issues, configuration items, IVV procedures, etc.
IE Object Prefix	string	see description	Used to set the Project Unique ID attribute within formal modules. Note that the PUID is calculated with the following rule : [module prefix]-[object prefix]-[number] - the [module prefix] is obtained from the DOORS module attribute "Module Prefix" of the formal module, - the [object prefix] is obtained from this attribute ("IE Object Prefix"), for the appropriate object type - the [number] is incremented by one every new Object. Thus for a given Object Type, the object prefix is the same for the whole project, as it is defined once in the PP module
IE Object Word Style	string	see description	This attribute defines the paragraph style to applied to each RMF object when it is created. This style is used if the object is exported to Word, and a style definition with the same name is found in the Word template.

The following attributes apply to the definition of modules, and are set to null in other objects

Attribute name	Type	value	Description
IE Module Type	string	see description	Represents a module type usable in the current project. The objects with this attribute set to a value (i.e. not NULL) describe the data definition of modules suitable to build the project structure. The modules contain the objects. Typical modules are User Requirements, System Requirements, Issues and Decisions, PBS, IVV Procedures, etc.
IE Module Word Styles	text	see description	This attribute is used to record the list of possible Document (ie Paragraph) styles which may be applied to objects within the given Module Type. Each line of this text attribute represents a style.
IE undefined PUID keyword	string	See description	Contains the keyword to set in the PUID attribute when the Manage objects tool is used in a particular context: module open in edit shareable mode. PUIDs can only be assigned in Exclusive Edit mode, so in the Edit Shareable mode context it is assigned to the default value set by this attribute. It can be an empty keyword. The “Renumber object” tool can complete the PUID later, but it is recommended that it is set to a meaningful reminder, such as “PUID TBD later”
IE Bitmap	string	A bitmap filename with a .bmp extension	Name of a file containing a bitmap associated with the current module type and which will be displayed in the Relationship Manager.

The following attributes apply to the definition of relationships, and are set to null in other objects

Attribute name	Type	value	Description
IE Relationhip	string	see description	Represents the name of a link module It must be one of the link module available in the current project.
IE Source Module	string	see description	Represents the source module of an authorized relationship. It must be a module type usable in the current project. Typical values are User Requirements, System Requirements, Issues and Decisions, PBS, IVV Procedures, etc.
IE Target Module	string	see description	Represents the target module of an authorized relationship. It must be a module type usable in the current project. Typical values are User Requirements, System Requirements, Issues and Decisions, PBS, IVV Procedures, etc.
IE Mapping	string	Many-to-Many, Many-to-One, One-to-Many, One-to-One	Describes the authorized cardinality of the relationship

The following attributes apply to the definition of templates, and are set to null in other objects

Attribute name	Type	value	Description
IE Template	string	see description	Represents the name of a template (a module in the folder Module Types).
IE Attribute Name	string	see description	Represents the name of a semantic attribute defined into the template. Each semantic attribute is described by an object with IE Attribute Name not NULL in the Template object
IE Is Semantic Attribute	bool	True,False	True for an attribute to be controlled by RCM
IE Is Volatile Attribute	bool	True, False	True for an attribute to be controlled by RCM, but that is reinitialized when creating a new version
IE Is Contextual Attribute	bool	True,False	True for an attribute managed by PFM

The following attributes apply to the definition of views, and are set to null in other objects

Attribute name	Type	value	Description
IE View Name	string	see description	Name of a model level Explorer view
IE View	text	see description	Description of the Explorer view.

Available relationships for Project Profile objects

None

Available views of Project Profile module

View name	Filter	Description
standard view	None	DOORS default view, which shows the DOORS ID and Object Heading/Text.
Definition of the module types	IE Module Type != NULL	Displays the attributes: Object Text, IE Module Type, IE Module Document Styles and IE Bitmap.
Definition of the object types	IE Object Type != NULL	Displays the attributes: Object Text, IE Object Type, IE Object Prefix, IE Object Document Style and IE undefined PUID keyword.
Definition of the Relationships	include object 41 show descendants	Displays the attributes: Object Text and Heading, IE Relationship, IE Source Module, IE Target Module, IE Mapping.
Definitions of the Templates	IE Template != NULL and IE Module Type == NULL	Displays the attributes Object Text,,IE Template, IE Attributer Name,IE Is Semantic Attribute,IE Is Volatile Attribute, IE is Contextual
Definition of the views		Displays the attribute Object Text, IE View Name, IE View
Project Data Model representation	include object 17 show descendants	Displays the picture showing a graphical representation of the project data model.

Appendix G. FREQUENTLY ASKED QUESTIONS (FAQ)

Why it is impossible to identify a RMF object within a table ?

Because this is not convenient and practical to manage data in tables with DOORS :

- No direct display of the attributes of cells (no analysis view),
- Table can't be displayed in traceability matrix (DXL column),
- DOORS does not allow the selection of multiple cells or a row, so it's not possible to identify such requirements or make join operations. DOORS only allows the selection of one cell or the complete table.

  Note : A complete table can be identified as a requirement.

I specified colors for my enumerated attribute and they are not displayed in the views. Why ?

Check if in the column properties, the option "Color by attribute" is set. Don't forget to save the view.

How can I change the color of the text of RMF Objects in a view to distinguish them from ordinary text ?

First, define colors for the type "IE Object type" and then in the text column of a view, edit the column properties in order to set the option "Color by attribute" selecting the attribute "IE Object type". Don't forget to save the view. Notice that the chosen enumerated attribute must not be multi-valued.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send written license inquiries to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send written inquiries to:

IBM World Trade Asia Corporation

Licensing

2-31 Roppongi 3-chome, Minato-ku

Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS"WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions. Therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it

believes appropriate without incurring any obligation to you. Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Intellectual Property Dept. for Rational Software

IBM Corporation

1 Rogers Street

Cambridge, Massachusetts 02142

U.S.A.

Such information may be available, subject to appropriate terms and conditions,

including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, ibm.com, Rational, DOORS, are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both, are trademarks of Telelogic, an IBM Company, in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at www.ibm.com/legal/copytrade.html.

Adobe, the Adobe logo, Acrobat, the Acrobat logo, FrameMaker, and PostScript are trademarks of Adobe Systems Incorporated or its subsidiaries and may be registered in certain jurisdictions.

Microsoft, Windows, Windows 2003, Windows XP, Windows Vista, Excel and/or other Microsoft products referenced herein are either trademarks or registered trademarks of Microsoft Corporation.