

ReqXChanger

Closing the Gap between Requirements Management
and Model Driven Software Engineering

WILLERT.



The Gearwheel between Requirements and UML/SysML

An often untapped opportunity to increase the efficiency is to avoid or to close gaps between different engineering tools. Such a gap has always many disadvantages. First of all, there is a broken information chain. Therefore, somebody is needed to maintain data consistency in different tools. A thankless, time-consuming and error-prone task for a human being. Moreover, there is always uncertainty about: "is this data up to date?"

What we need is an interface between different engineering tools. An interface that provides easy data synchronization. An interface that draws attention to synchronized changes. Willert Software Tools has created such an interface between common requirements management tools and UML / SysML modeling solutions – IBM Rational Rhapsody and Enterprise Architect. This interface is called the ReqXChanger.

The ReqXChanger can read requirements from files in the standardized ReqIF format and creates representations in the UML/SysML modeling tools for them. This allows to link model elements to requirements directly inside the modeling tools. In addition, links to between model elements and requirements are analyzed by the ReqXChanger and it can create another ReqIF file, which contains information about all linked model elements. This ReqIF file can be imported into a requirements management tool, in order to achieve a full traceability inside it.

Furthermore, when frequently synchronizing requirements with the modeling tools, changed requirements are highlighted. This enables to make an impact analysis.

Synchronization of Requirements with UML/SysML

For instance between:

DOORS and Enterprise Architect
Polarion and Rhapsody

Standardized Format

Any Requirements Management Tool with ReqIF can be used

Synchronization of Model Elements

Synchronization of Model elements that are related to Requirements between UML/SysML and Requirements Management

UML Diagrams in Requirements Management

Transfer UML diagrams that are related to requirements to Requirements Management tools

Suspect Links

Highlights changed requirements in UML/SysML

Full Traceability

Enables to establish a full traceability between Requirements and UML

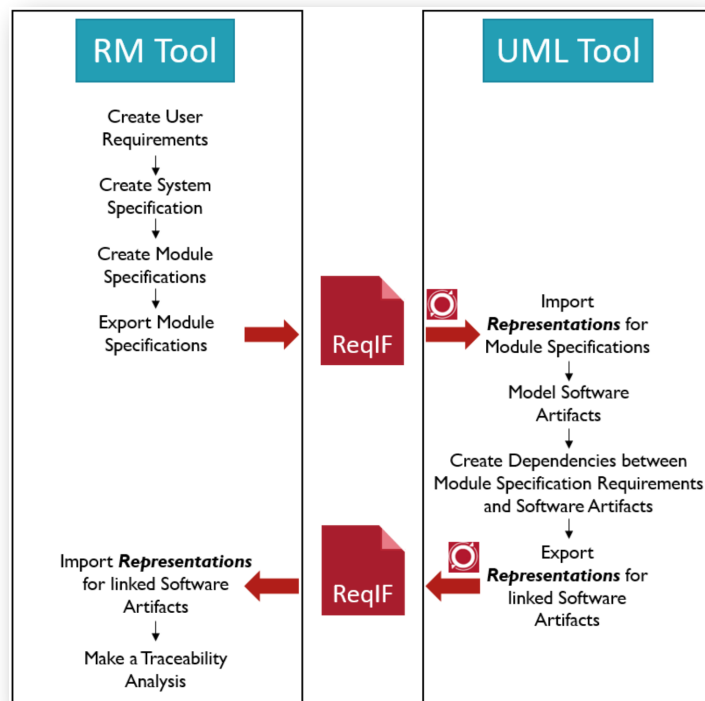
Automatization

Exchange can be automated

Exchange Requirements using ReqIF

Inside a requirements Management Tool, you will usually create requirements on different levels. These levels can be for *User Requirements*, *Design Requirements* and *Module Specifications*. The connection between these levels is realized by creating *links* from requirements on a lower level to requirements on the level above. A requirement on a lower level *satisfies* the linked requirement above. So, Module Specifications satisfy Design Requirements and Design Requirements satisfy User Requirements.

Module Specifications are the only requirements here which are directly satisfied by Software Artifacts, which you create inside a modeling tool e.g. Rhapsody. Therefore, only these have to be transferred to it.



After exporting a ReqIF file from a Requirements Management Tool with Module Specifications, you can use the ReqXChanger to create *representations* for the requirements inside a UML tool. The Requirements Management Tool is considered to be the *master* of these requirements. This means, that this is the tool for modifying the requirements and to create new requirements. In the UML tool, there are only representations of these requirements, which are not expected to be modified. While developing a software with a UML tool, you will create several model based software artifacts. For creating a connection between these artifacts and the Module Specification requirements, Dependencies are used. A Dependency has the software artifact as source and the requirement as target. Furthermore, a Stereotype is assigned to the Dependency to define its kind.

Then, the ReqXChanger can be used to export *representations* of software artifacts, which have a Dependency to a requirement. For the export, the same ReqIF file is used as for transferring the requirements to the UML tool. Afterwards, the ReqIF file can be used to import the representations of the software artifacts into the Requirements Management Tool. Furthermore, Links are created as representations for the Dependencies. As a Requirements Management Tool is the master for the requirements, the UML tool is considered to be the *master* for software artifacts and Dependencies.

Now you can make a complete traceability analysis inside your Requirements Management Tool.

Since UML tools have a different model structure than Requirements Management Tools, the requirements with their information have to be mapped to other elements. Therefore, the presented table shows the mapping of requirements and other elements from a Requirements Management Tool to UML tools. As examples for Requirements Management Tools, DOORS and Polarion are used in this handout.

The ReqXChanger creates a Package for a Module / Document. Package names can be configured inside the options of the ReqXChanger. For a Requirement, requirements are created and Tags are used to store attribute values. Stereotypes are created for Requirements Types and assigned to requirements.


Requirements Management	UML tool
Document	Package
Requirement	Requirement
Attribute Values	Tags with Values
	Comments for additional ReqIF Attributes
Requirement Type	Stereotypes
Datatypes	Datatypes (not for Primitive Types)




Transfer Requirements to UML/SysML

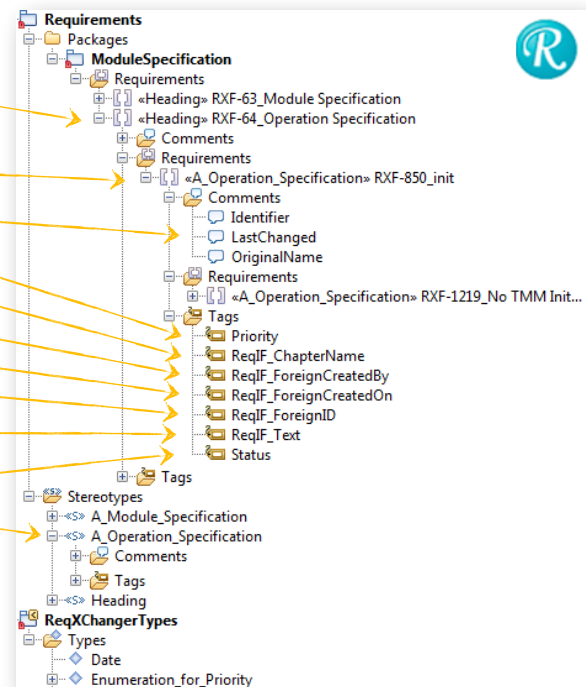
2 Operation Specification

RXF-850 - init()

Initializes the RXF by calling init() functions of sub-modules of the RXF (MEM, TMM, MSQ) in the correct order.



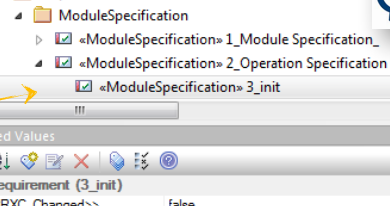
Updated	2014-02-19 15:31
Priority	 recommended [50.0]
Title	init()
Author	Eike Römer
Created	2013-04-29 10:55
ID	RXF-850
Description	Initializes the RXF by calling init() functions of sub-modules of the RXF (MEM, TMM, MSQ) in the correct order.
Status	 Draft
Type	 A Operation Specification



As an example, a graphic shows how a requirement from a document inside Polarion is transferred to Rhapsody. The document contains requirements of the Realtime Execution Framework (RXF) from Willert Software Tools. For each ordinary attribute value, a tag is created which contains the value. Furthermore, there are some comments, which also contain attribute values. The difference is that the comments are used for attributes, which directly exist for requirements inside ReqIF files. As an example, there is a unique identifier, which is not the same as the identifier inside Polarion.

The name and the description of requirements inside Rhapsody can be mapped to any attribute value. This can be configured inside the export options of the ReqXChanger.

		Last Modified On
1	Module Specification	Montag, 16. November 2015
2	Operation Specification	Montag, 16. November 2015
3	2.1 init() Initializes the RXF by calling init() functions of sub-modules of the RXF (MEM, TMM, MSQ) in the correct order.	Montag, 16. November 2015

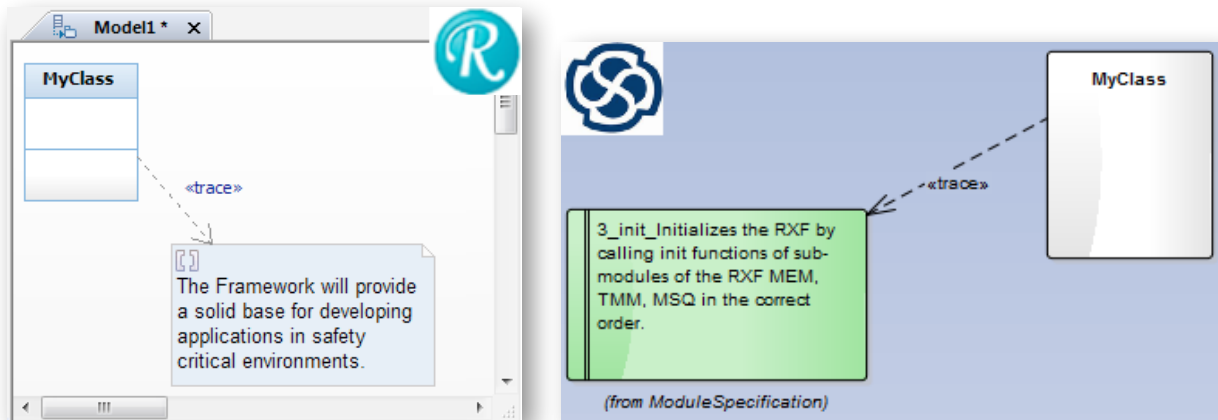


The screenshot shows the Blinky IDE interface. At the top, the 'Requirements' view is open, displaying a tree structure under 'ModuleSpecification'. Three items are listed: «ModuleSpecification» 1_Module Specification, «ModuleSpecification» 2_Operation Specification, and «ModuleSpecification» 3_init. The '3_init' item is selected. Below the Requirements view, the 'Tagged Values' section is visible, showing a table of properties for the selected requirement.

Property	Value
<<RXC_Changed>>	false
<<RXC_Deleted>>	false
ReqIF_ChapterName	init()
ReqIF_ForeignCreatedBy	Administrator
ReqIF_ForeignCreatedOn	2015-11-15T23:00:00.0Z
ReqIF_ForeignCreatedThru	Manual_Input
ReqIF_ForeignID	3
ReqIF_ForeignModifiedBy	Administrator
ReqIF_ForeignModifiedOn	2015-11-15T23:00:00.0Z
ReqIF_Identifier	_3_026aac4f4fb5-4ddf-8a8a-23a1ed997b3e
ReqIF_LastChanged	2015-11-16T17:47:22.0Z
ReqIF_LongName	
ReqIF_Text	Initializes the RXF by calling init() functions of s...

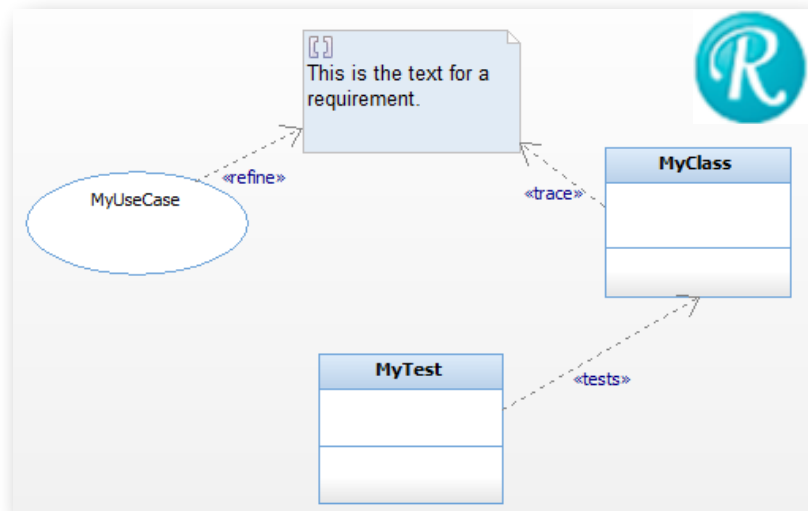
As a second example, a graphic shows how requirements from DOORS are transferred to Enterprise Architect. The Heading of a requirement is mapped to the attribute *ReqIF_Title* and the text is mapped to the attribute *ReqIF_Text*. All the attributes are stored as tagged values of requirements in Enterprise Architect.

Establish Links from Model Elements to Requirements directly in UML/SysML

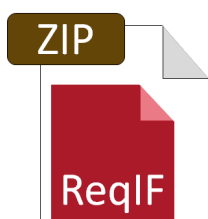


Inside Rhapsody and Enterprise Architect, links can be created from model elements to requirements using Dependencies. This can be performed in a model browser or inside diagrams. In addition, stereotypes can be applied to the Dependencies for distinguishing between different kinds of links.

Any model elements like use cases, classes, operations, attributes, states and diagrams can be linked to requirements. Furthermore, indirect links can be created, for instance between a test case and a Class, which has a Dependency to a requirement.

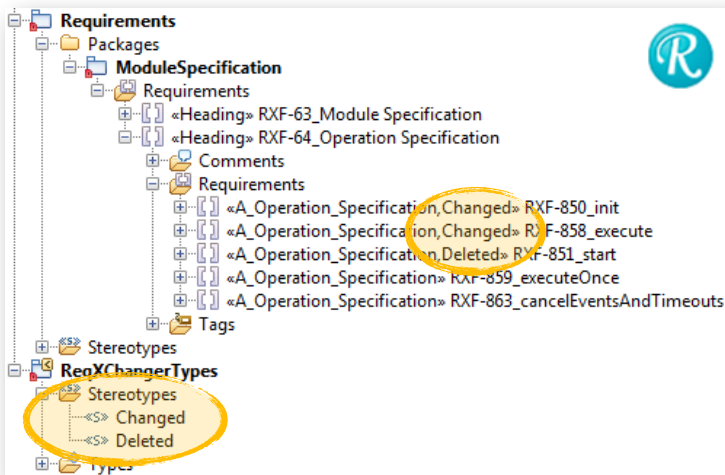


Support for ReqIFz (ReqIF-Zip) Files



The ReqXChanger supports to work with a ReqIF file, that is compressed inside a ReqIFz (Zip) file. This file can not only contain one document from a requirements management tool, but several documents at once.

Traceability Analysis and Suspect Links

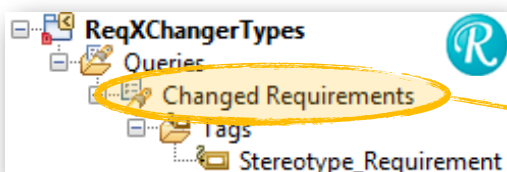
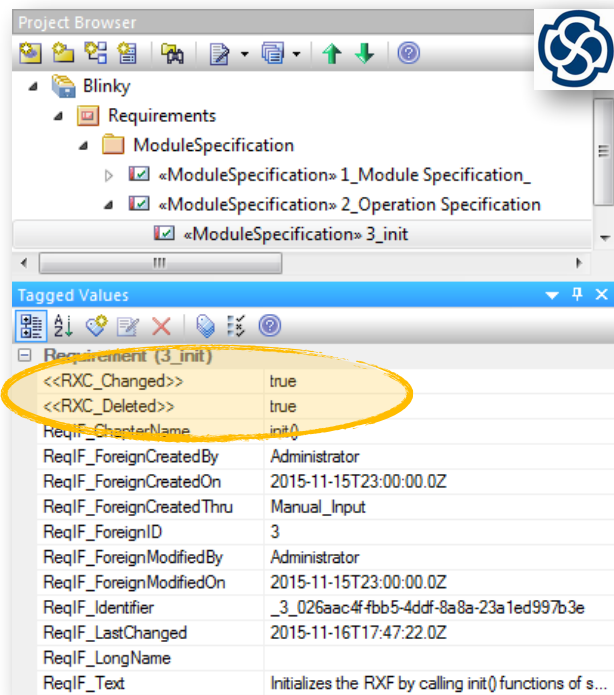


In order to make a *traceability analysis* inside your Requirements Management Tool, representations of modeled software artifacts are transferred back to the RM Tool. Therefore, a new Requirements Type (Work Item Type) is created with attributes for the *name, type, path in packages and a unique identifier*. The representations of the software artifacts are created as requirements (Work Items) of this type and links are created to requirements, corresponding to the trace Dependencies inside Rhapsody.

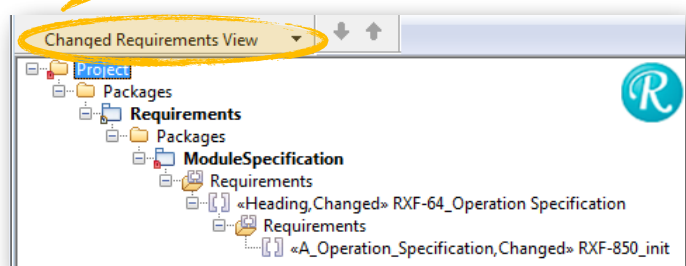
When requirements are changed inside your RM Tool, you can update their representations inside Rhapsody by re-transferring them. If a requirement has changed or was deleted since the last transfer, the stereotype *Changed* or *Deleted* is assigned to the requirement. You can use the *Model Browser* of Rhapsody to *filter* for these requirements and *investigate suspect links*.

In Enterprise Architect, the suspect link functionality is slightly different. Here, additional tagged values `<<RXC_Changed>>` and `<<RXC_Deleted>>` are used to indicate, if a requirement has changed or if it was deleted. Then, these values are set to "true".

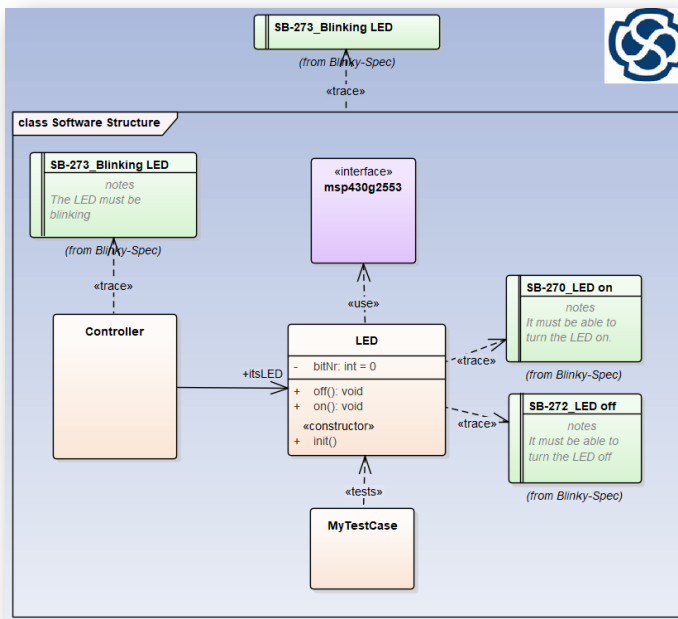
In order to show all changed and/or deleted requirements, Enterprise Architect offers to configure search queries, which can search for requirements that have these tagged values set to "true".



As an example for filtering changed requirements, a query is defined here in Rhapsody for searching for them. Besides executing the query by double clicking on it, it can also be assigned to the model browser as depicted here.



Transfer Model Elements and Diagrams to Requirements Management Tools

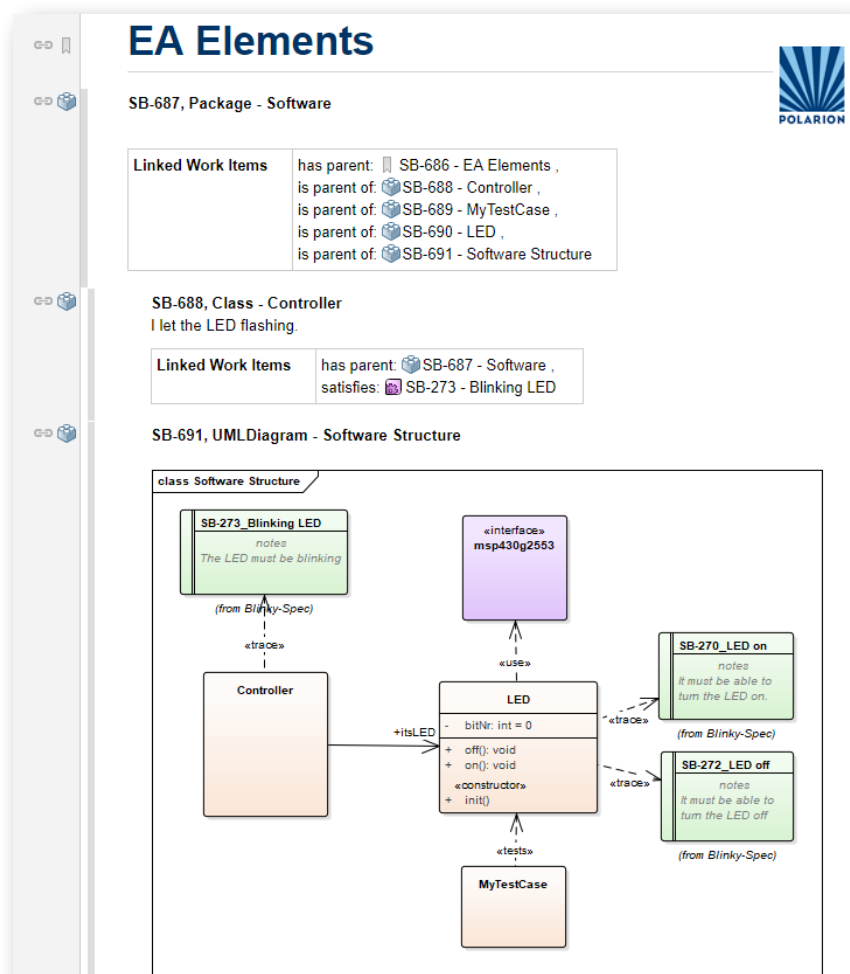


Information about all model elements, which have a Dependency from it to a requirement, can be transferred to the requirements management tool. This is also valid for diagrams that are linked to requirements.

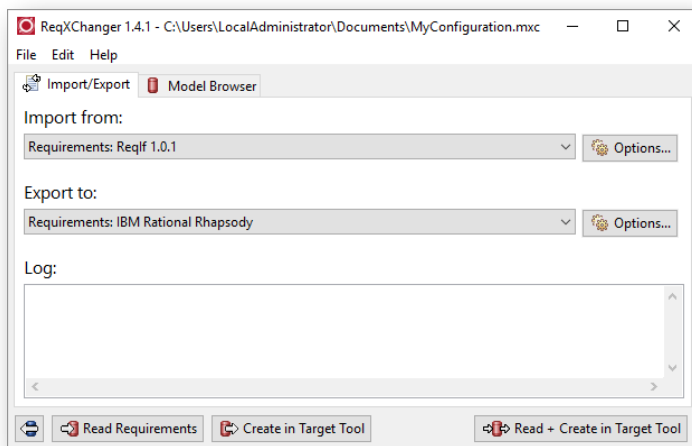
In order to perform this, the ReqXChanger will go through all the requirements inside the original ReqIF file and searches for their representations inside the modeling tool. For these, it will check if there are incoming Dependencies from other modeling elements. If this is the case, it will create elements inside the ReqIF format as representations of the model elements together with links to the requirements. This information is saved inside a new ReqIF file, which can be imported inside the requirements management tool. For diagrams, it will also save a screenshot besides the new ReqIF file.

Importing the new ReqIF file with representations for model elements into the requirements management tool will result in a new document for the model elements.

For model elements, not only their name and description becomes visible inside the document, but also additional information like their type (e.g. Class or Package), path in the model and their Identifier in the modeling tool. For diagrams, the created screenshot will also be visible.

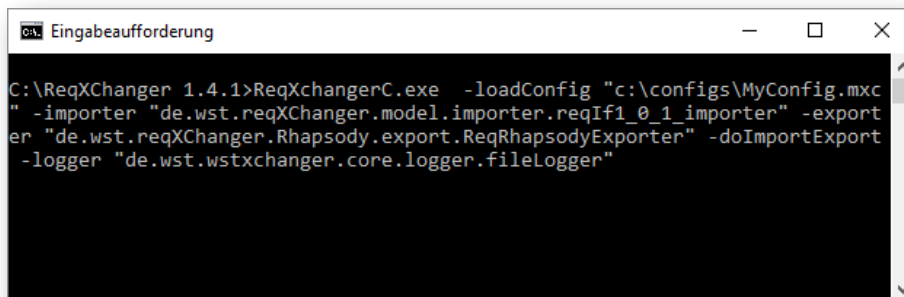
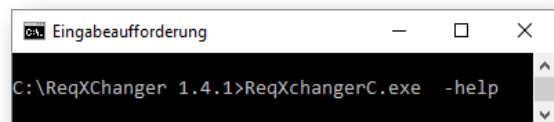


Different Ways of Execution and Automatization



The ReqXChanger has a GUI for selecting the source of the requirements and the target modeling tool. For these, several options can be configured and the exchange of information in both directions can be performed by pressing corresponding buttons. Furthermore, a logging segment will show information about the data transfer and a Model Browser allows to investigate an imported ReqIF file.

However, it may not always be desirable to execute the data transfer manually. Therefore, the ReqXChanger is highly automatable. Any functionality of the ReqXChanger can be invoked using command line arguments. This does not only include to select a target tool and execute the data transfer, but also to set option values. This can be done either by setting all option values using parameters or by loading a configuration file. In addition, different logging mechanisms can be set, for instance for logging inside a log file. In order to show all possibilities of the ReqXChanger, execute `ReqXChangerC.exe -help` (ReqXChangerC.exe will write on the console).



As an example, here it is shown how to start the ReqXChanger for loading a configuration file, transferring requirements to Rhapsody and perform logging inside a log file.

Full Requirements Traceability, even in Source Code

```
// Realizes requirement RXF-850_init
//## operation init()
void init();
```

By creating the Dependencies between model elements and requirements, the traceability does not end in the model, but goes down to the source code. Code generators, which can automatically create the source code for your model, can also generate information about linked requirements inside the source code.



WILLERT.

Available webinars and trainings about this topic:

Traceability zwischen Anforderungen und UML-Modell

Actual dates see: www.willert.de/Termine

Related products:

IBM Rational Rhapsody Gateway

Author:

ARNE NOYER

JOACHIM ENGELHARDT

Contact:

WILLERT SOFTWARE TOOLS GMBH

Hannoversche Straße 21

31675 Bückeburg

www.willert.de

info@willert.de

Tel.: +49 5722 9678 - 60