CRYSTAL

CRitical S**YST**em Engineering **A**cce**L**eration

knowledgeMANAGER

D607.041

DOCUMENT INFORMATION

Project	CRYSTAL
Grant Agreement No.	ARTEMIS-2012-1-332830
Deliverable Title	knowledgeMANAGER
Deliverable No.	D607.041
Dissemination Level	CO
Nature	P
Document Version	V3.00
Date	2013-11-29
Contact	José Fuentes
Organization	REUSE
Phone	+34912172596
E-Mail	jose.fuentes@reusecompany.com

AUTHORS TABLE

Name	Company	E-Mail
José Fuentes	REUSE	jose.fuentes@reusecompany.com

CHANGE HISTORY

Version	Date	Reason for Change	Pages Affected
1.0	30.Oct.13	Initial version	All
2.0	8.Nov.13	Changes after the internal review	Minor changes along the document
3.0	28.Nov.13	Changes after the external review	Brief introduction to the WP607. Clarifications to the architecture figure. Change in the picture of the ontology layers. Added a picture depicting different roles

CONTENT

Content of Figures

Content of Tables

1 Introduction

1.1 Role of deliverable

The goal of this deliverable is to describe the features included in knowledgeMANAGER (kM), brick (B2.37).

This brick is part of WP607. The main goals of this WP are the following:

- CCC approach for quality requirements:
  - Correctness – Consistency - Completeness
- Requirements reuse
- Integration with IOS
- Support to the development supply chain and shareable content

As a result of the main objectives of the WP, the main goal of this brick is managing all knowledge needed to measure, manage and improve the quality of every system artefact eventually managed through RQS. In the case of Requirements engineering, knowledge deals with managing the quality of requirements specifications. For that purpose, knowledgeMANAGER mainly deals with Natural Language (NL) processes, thesaurus and requirements patterns, all of them integrated in the System Knowledge Base (SKB) represented as an ontology.

1.2 Relationship to other CRYSTAL Documents

This deliverable is related to the rest of deliverables and bricks created in WP6.7 as well as the main deliverables from UC2.4 (Electrical Flight Control System - the UC which WP6.7 is based on). The level of relationship is the following:

- D607.011: since this document will describe the user needs and goals, the development and finally the assessment of the brick knowledgeMANAGER (B2.37)
- D607.021: since the ontology and requirements patterns managed by knowledgeMANAGER are used in RQA to accurately improve the quality of a requirements specification (brick 2.29)
- D607.031: since the ontology and requirements patterns managed by knowledgeMANAGER are used in RAT to provide a writing assistant (brick 2.30)
- D204.013: since this deliverable represents the needs of the industrial partners involved in WP6.7 (UC2.4).

1.3 Structure of this document

The structure of the document is the following:

- Chapter 2 – Current technical features: first we start with the description of the current state of the brick
- Chapter 3 - Training offered to end-users: this chapter lists the training sessions that have been held related to this brick, a link to the training material is also included
- Chapter 4 – Main goals for the brick during the CRYSTAL project: finally, this chapter summarizes what seems to be the most important goals for the industrial partners related to the WP

2 Current technical features

This chapter describes the current technical features of the knowledgeMANAGER tool. As a first iteration for this brick/deliverable, the set of features described hereinafter correspond to the status of the tool as it is today in the commercial version (version 6.1), available at http://www.reusecompany.com.

2.1 Description of the tool

knowledgeMANAGER (kM) belongs to Requirements Quality Suite (RQS), a set of tools aimed to customize, manage and improve the quality of a set of requirements (see also deliverables D607.021 and D607.031 for a more detailed description of the other tools included in the suite).

More specifically, the main goals of kM are the following (see section 2.3 for a more detailed description):

- Managing the indexing process based on Natural Language tools (NL tools)
- Manage controlled vocabulary to be used in RAT and RQA
- Manage thesaurus and links among the concepts in the controlled vocabulary
- Manage requirements patterns needed to generate the proper formalization of requirements (or any other text-based artefact)
- Manage the communication between the team in charge of creating the requirements and the team in charge of managing all the layers on the ontology. This communication is modelled as a suggestion system for the requirements authors.

All the previous information defines an ontology formed by the following layers:

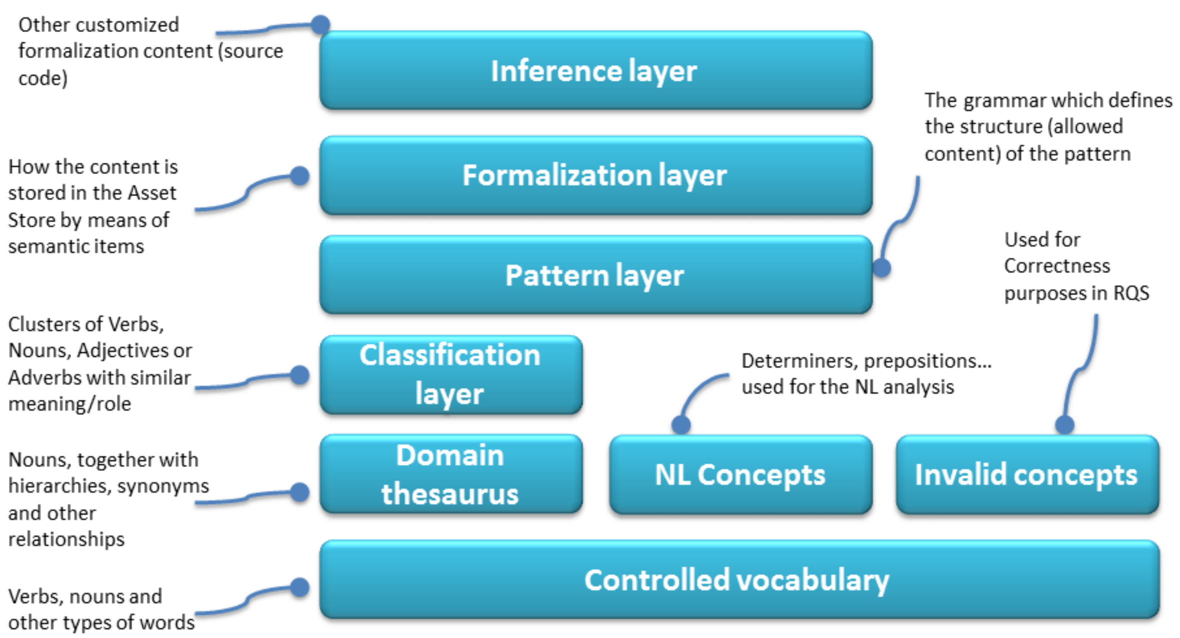

Figure 2-1: Ontology layers

Those layers will be described in section 2.3.

2.2 Architecture of the suite

This picture represents the architecture of the whole Requirements Quality Suite. The rest of the section describes all the boxes in the architecture and how and why kM is connected to the other tools.

This picture shows the dependencies among different components as blue arrows. Those components may or may not be installed in the same physical node (a Windows based computer), but all of them must be connected to the same LAN.

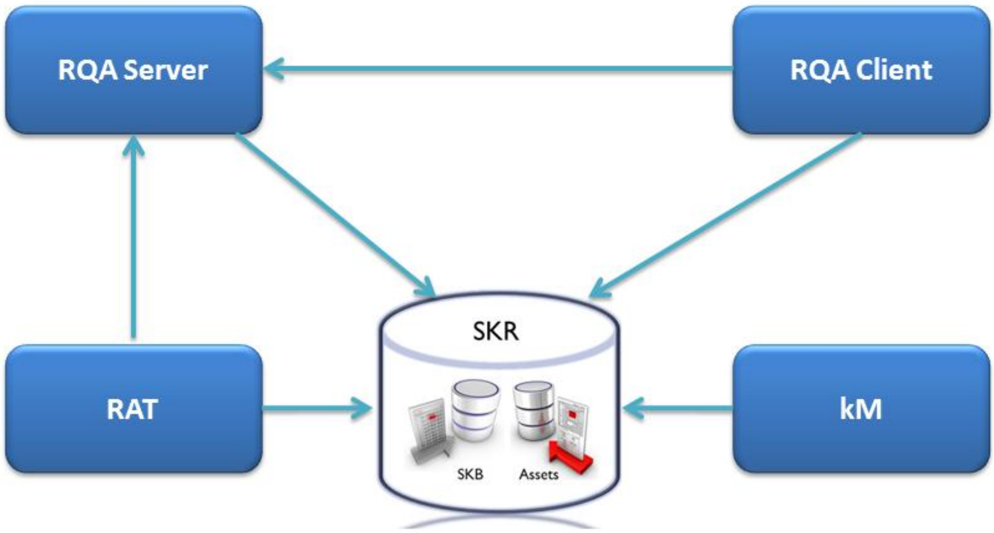

Figure 2-2: RQS Architecture

The components of this architecture are the following:

- RQA Server Requirements Quality Analyzer Server:
  in charge of the main configuration of the whole suite. Database connection, licensing and low-level database management…

- RQA Client – Requirements Quality Analyzer Client:
  provides the customization of the quality assessment of the suite. RQA is using the terminology provided in the various layers managed by knowledgeMANAGER, as well as the rest of the layers of the ontology managed by kM

- RAT – Requirements Authoring Tool:
  this module allows quality analysis on the fly, but it also includes the authoring guidance by using the information represented in the boilerplate layer

- kM – knowledgeMANAGER:
  fully described in this document

- SKR – System Knowledge Repository:
  this is a relational database where we can find two clearly different parts:
  - SKB – System Knowledge Base: represents the main ontology behind all the quality analysis as well as all the information needed to perform a Natural Language Process to generate a semantic graph out of a textual requirement
  - Assets: represents the formal representation (mainly as a semantic graph) generated out of every textual requirement once the requirement has been created with RAT

2.3 List of features

2.3.1 Introduction

The main goal of kM is to manage all the knowledge needed by RAT and RQA (or by any other further tool which projects/documents could be semantically indexed) to be able to measure the quality of a requirement specification.

kM includes the following features:

- Vocabulary management: including screens to manage the controlled vocabulary of a project
- Thesaurus construction: including features to semi-automatically creating thesaurus
- Thesaurus management: including screens and techniques to manually build thesaurus
- Classes management: allows to create and manage classes for Terms and relationships
- Requirements patterns management: including screens to manage patterns and the way they are able to formalize (represent) the knowledge of the requirements

2.3.2 Support for the Natural Language Process

2.3.2.1 Introduction to the NL process

The first goal of RQS, when a requirement is analysed, is to generate a semantic graph out of the textual content of the requirement. An example of this kind of semantic representation is depicted in the following figure.

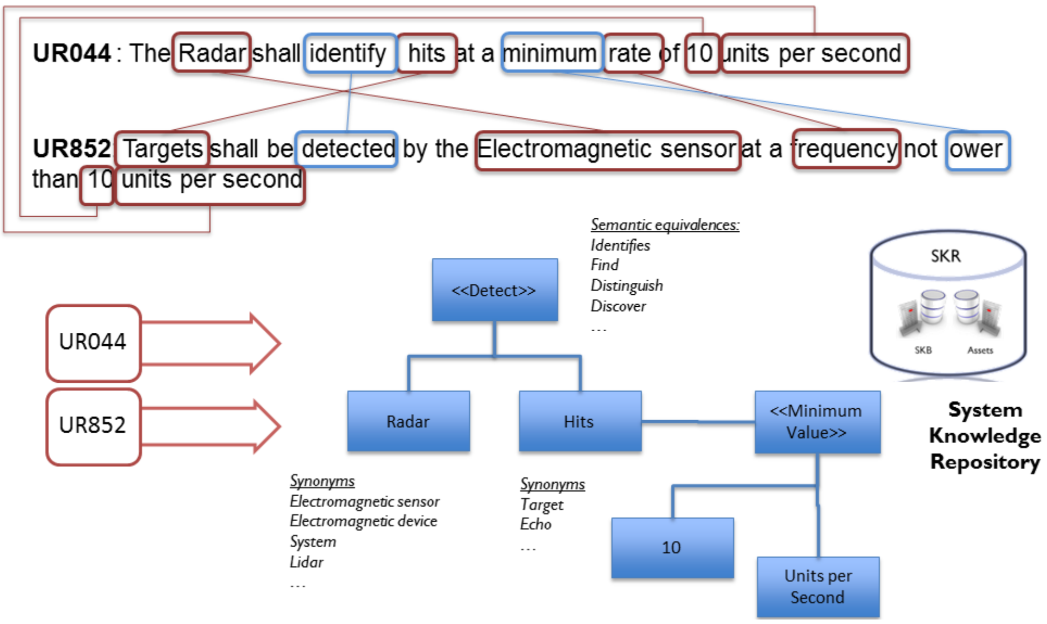

Figure 2-3: Semantic formalization

This figure shows, in blue, a semantic graph. In order to generate such a semantic graph, the following information is needed:

- A thesaurus including a list of concepts (controlled vocabulary) together with a set of links among the concepts, synonymy…
- A classification layer clustering together concepts with a similar meaning or similar purpose (e.g. two verbs with a similar meaning as *buy* and acquire or two concepts with a similar purpose such as two different stakeholders)
- A set of requirements patterns and their formalization, representing the “shape” and content of the final semantic graph

Using this information coming from the ontology, the following processes must be executed:

- Tokenization: this represents how the input text is split into individual words.
- Normalization: this stage deals with the morphology of the words, dealing with (mostly) the ending of the words in examples such us: singular/plural, verb conjugation, or even conjugation of irregular verbs
- Tagging: performed in those cases where the same word could have two different meanings, or could play a different role in the sentence. The word *plane* could be a noun, but also a verb and even an adjective. According to the structure of the sentence, the system must choose the most suitable tag (noun, verb, adjective…)
- Semantic grouping: some terms may have a particular meaning by belonging to one or more semantic clusters. This information is very valuable for further retrieval (coupling) processes
- Pattern matching: where different requirements patterns are evaluated to find the most suitable one matching with the input requirement
- Semantic formalization: once a pattern has matched with the input requirement, its formalization must be generated as a semantic graph
- Inference: allowing, by means of external libraries, to provide extra information after the standard formalization stage, or even to provide correctness, consistency and completeness information based on the already generated formalization of each requirement/artifact

2.3.2.2 Tokenization

Initially, every word is separated with spaces or punctuation marks. Nevertheless, sometimes the meaningful concept is represented by more than one word (compound concepts such as *breaking system*); and some other times, a “unit” of information (some characters enclosed into white spaces) shall be split into two or more tokens of information (e.g. 20Km/h is split into 20 as a number, and Km/h as a unit)

In order to join together two consecutive words, as a valid compound word, two options are available:

1. Including all compound words manually in the controlled vocabulary
2. Using the vocabulary identification screens to allow kM to generate all the possible compound words identified in a set of documents

In addition to that, a set of rules, based on regular expressions, can be used to properly split items as the example of 20Km/h explained above.

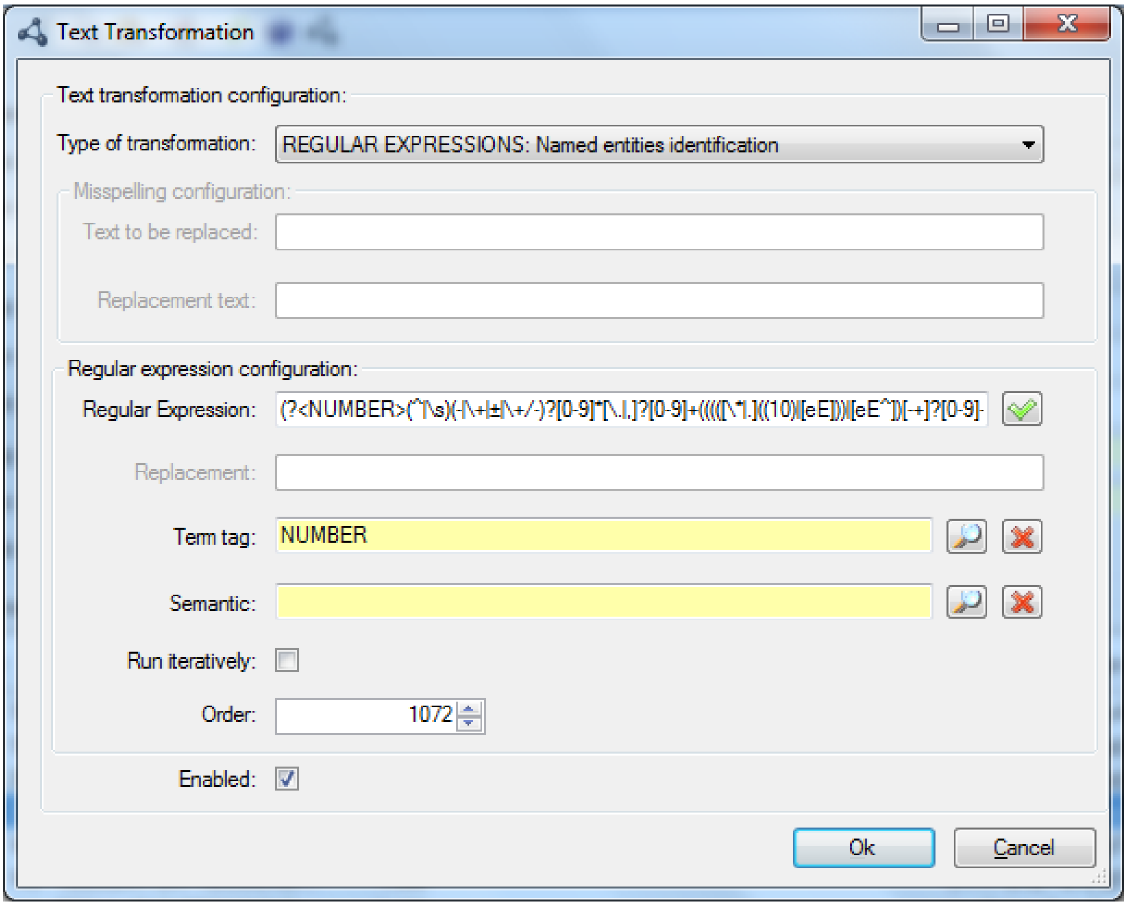

Figure 2-4: Tokenization

2.3.2.3 Normalization

A set of suffixes and substitutions rules are used for solving singular/plural, verb conjugation…

So, a rule to transform from "-ies" to "-y" will properly transform the input text "… flies …" into "fly" as the representative form of the verb *to fly*. kM can also manage irregular transformations , like "stood" for "stand".

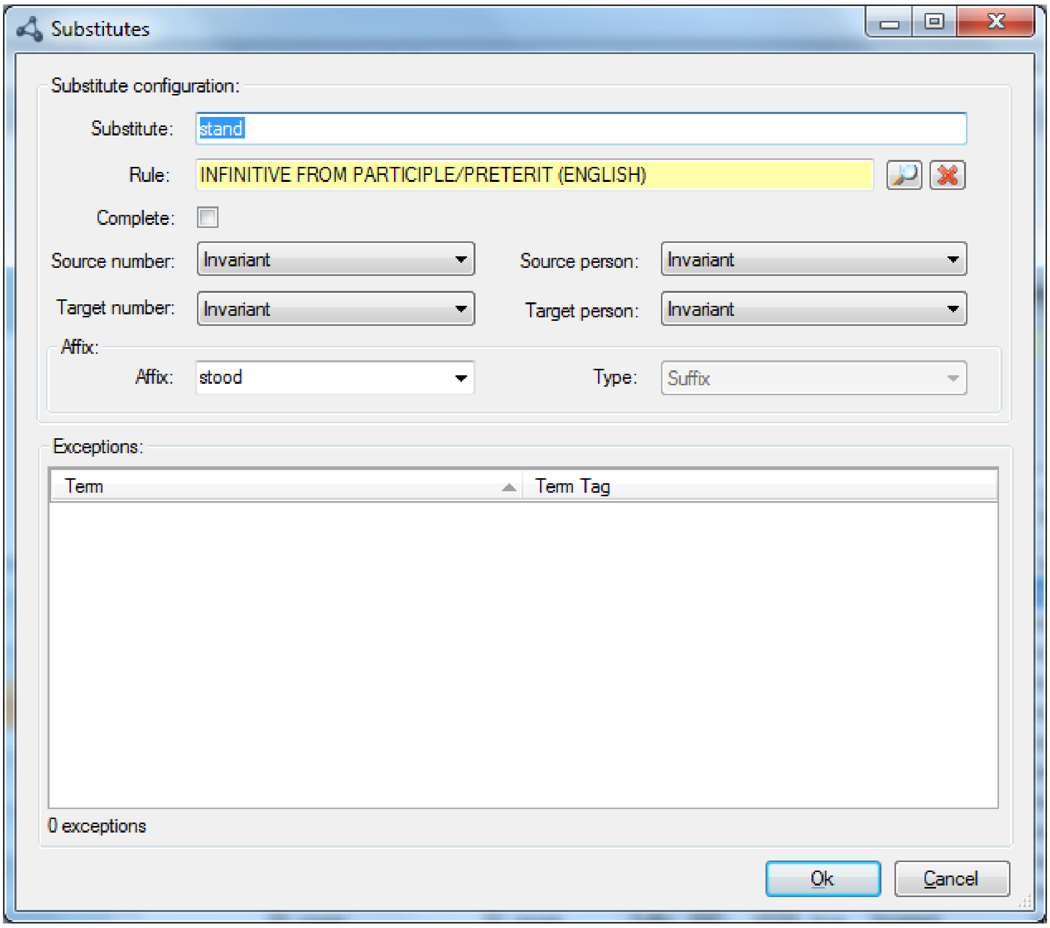

Figure 2-5: Normalization

2.3.2.4 Tagging

The goal of the tagging process is to take the output of the normalization process and find one (and only one) valid representative of this concept in the controlled vocabulary.

After the normalization step has been performed, three different possibilities can be faced:

1. Only one valid candidate is found in the controlled vocabulary: this is the most common case, and the unique candidate obtained after the normalization step is represented also after the tagging step

2. More than one valid candidate is found: here a unique candidate must be provided. In order to do that, a set of disambiguation rules is performed helping to decide which of the possible inputs (after the normalization step) is more likely to be the correct one.

   To do that, the context of the sentence is analysed, so that after realizing that "plane" can be either a verb, a noun or an adjective, if the sentences goes "the plane shall be able to…" this step must chose "plane" as a NOUN since its placed between a determiner and the shall verb, and this is the most likely choice

3. No valid candidate is found: the term will be included as an "unclassified noun". From this step on, this new concept shall be deemed as a special type of noun.

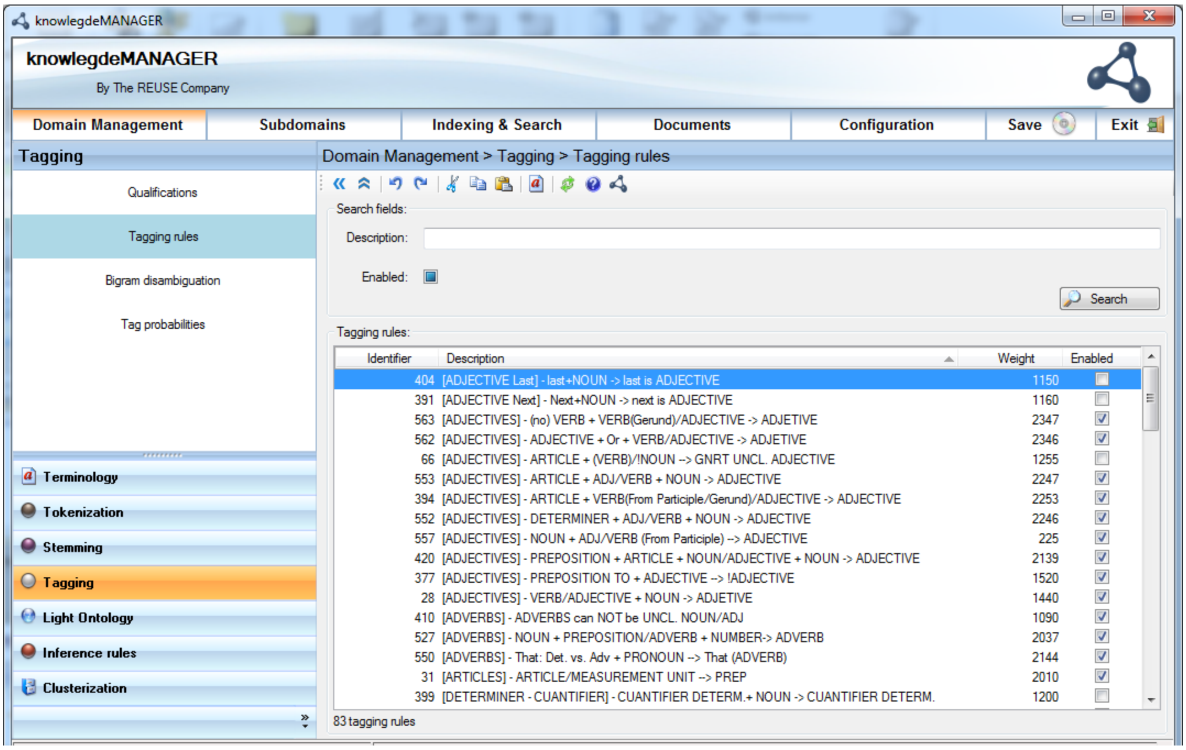

Figure 2-6: Tagging screen

2.3.2.5 Semantic grouping

knowledgeMANAGER is able to manage the definition of a set of semantics in order to cluster all the concepts in the controlled vocabulary into one or more of the clusters. The semantic grouping is, basically the support for the “CLASS-INSTANCE” relationship of an ontology and, as stated before, it’s used to group together concepts with similar meaning or similar purpose.

Assigning concepts into clusters can be done by using the semantics form (see figure bellow), or even from the vocabulary form.

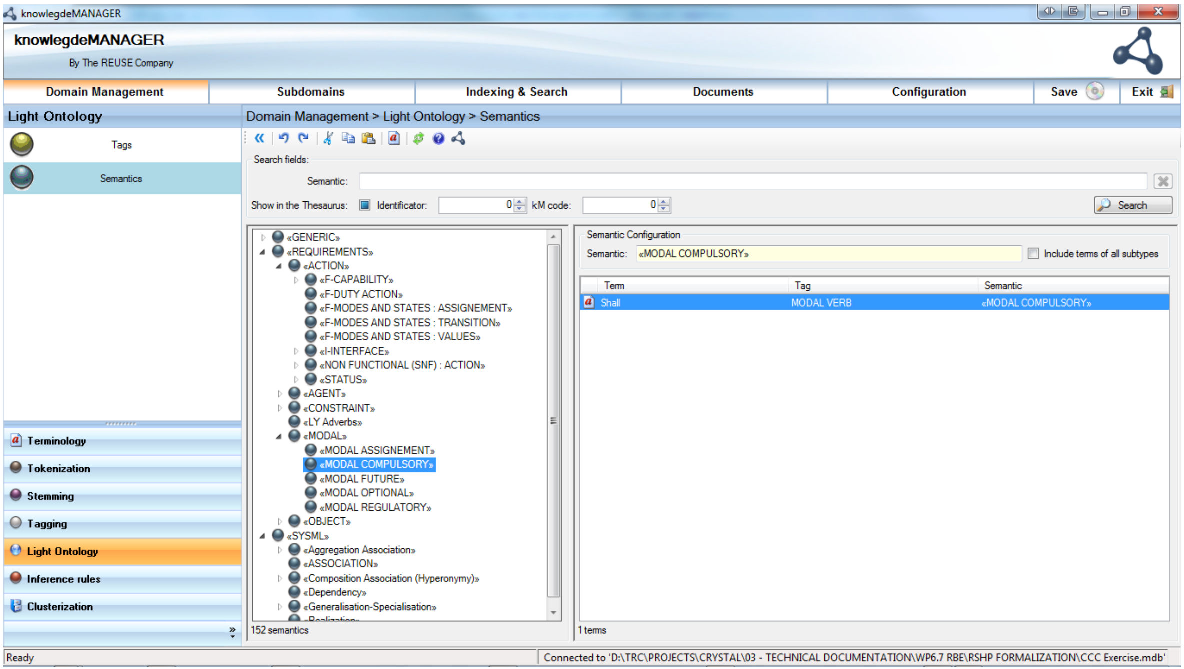

Figure 2-7: Semantic grouping

It's important to remark that semantic items can be categorized hierarchically, what is very valuable during the definition of requirements patterns. Thus, a semantic named <Actions> may have children such as <Transfer>, <Discover>…

2.3.2.6 Requirements patterns and formalization

See section 2.3.5.

2.3.3 Vocabulary management

This module allows the identification of the right set of concepts into the controlled vocabulary of a given knowledge domain. This process can be done manually (see Figure 2-8) or even following an easy process to identify new concepts automatically after analysing a set of relevant documents (see Figure 2-9); in the latter case, the frequency of occurrence of the concepts in the set of documents is a key point to decide whether or not a specific concept shall be part of the controlled vocabulary.

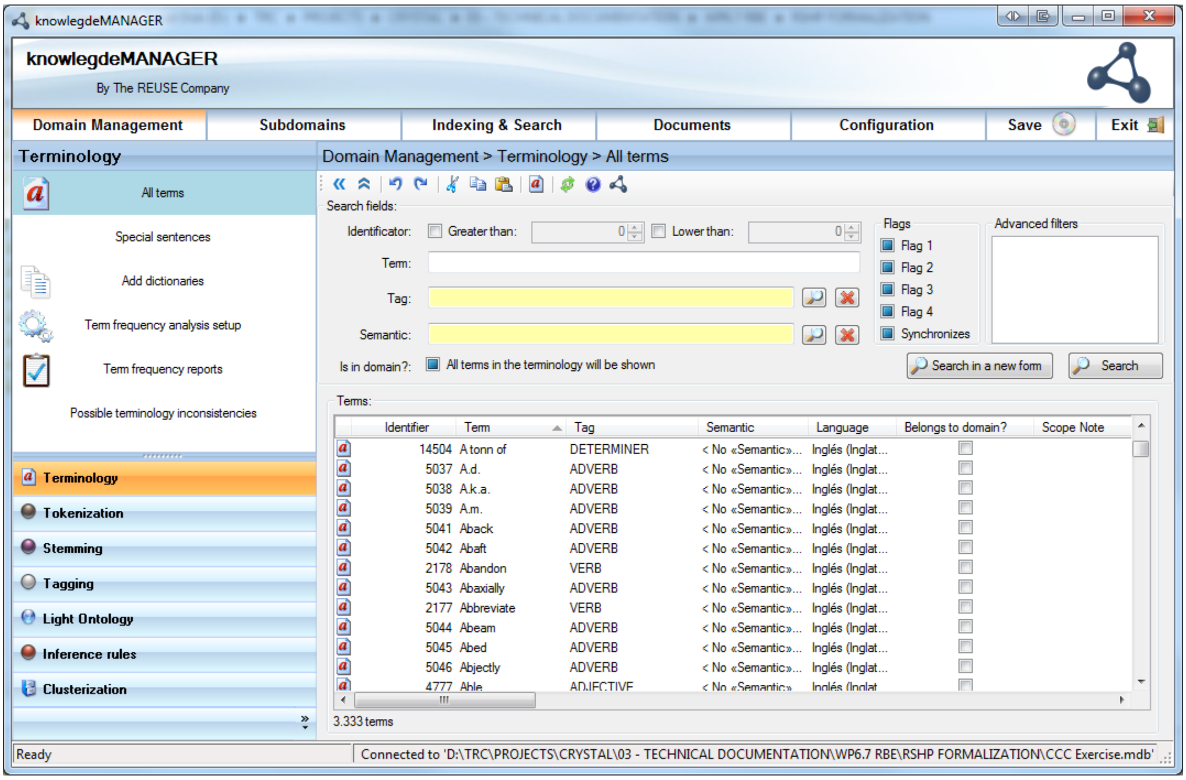

Figure 2-8: Vocabulary management

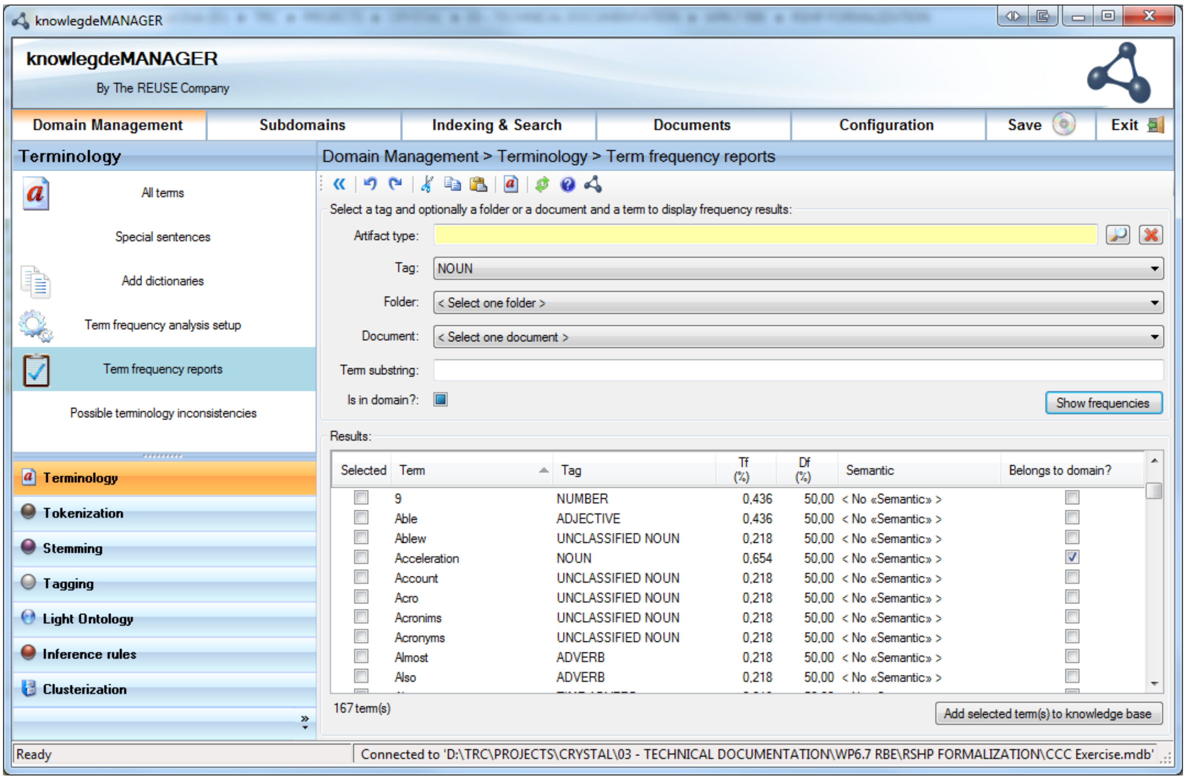

Figure 2-9: Automatic identification of concepts

2.3.4 Thesaurus management

A thesaurus allows adding different kinds of relationships to the concepts identified in the vocabulary management step. The most classical types of relationships are:

- Broader-Narrower: representing some kind of parent-child relationship as in the case of Stakeholder and Pilot
- Whole-Part: used to represent a physical breakdown as in the case of a PBS. An example could be Car and Braking system
- Related: represented other kind of dependency among concepts as in the case of Car and Driver

Those relationships will be used for later retrieval purposes, for naming suggestions, to check proximity among artifacts…

A very special type of relationship is the Synonymy, which is managed from a specific tab in the tool for every concept.

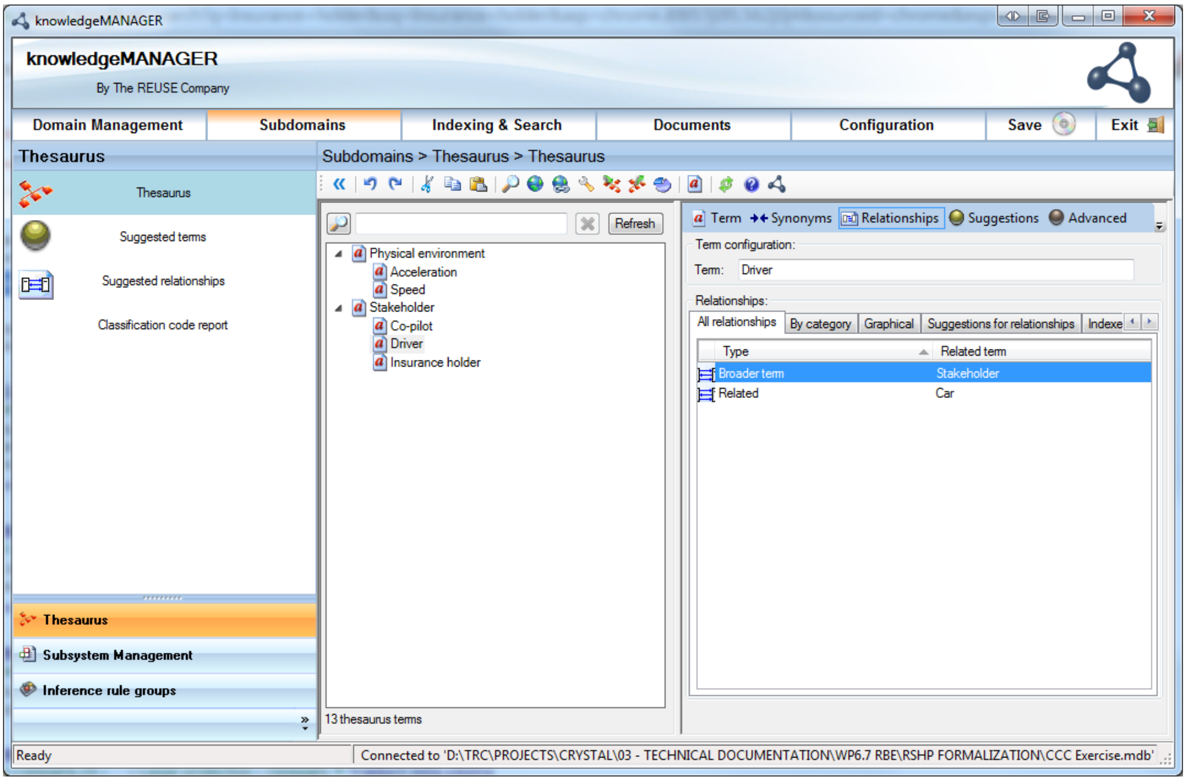

Figure 2-10: Thesaurus management

The picture above shows a small tree representing hierarchical or whole-part relationships; and also shows the whole set of relationships of the selected concept.

2.3.5 Requirements Patterns management

Requirements patterns can be considered as a sequence of different elements in a sentence, each of them could play a different role.

An example of pattern is the following:

While + in + **<mode>** + , + in case + the + **<component>** + to_be + **<trigger>** + , + the + **<component>** + must + **<action>**

Where many different requirements could match with them, for example the following:

While in landing mode, in case the button x is pressed, the emergency engine must start.

During the RBE process, requirements patterns are used to:

- Provide the proper formalization for further completeness and/or consistency analysis
- Provide the proper semantic formalization as a semantic graph for further retrieval and reuse stages
- Automatically identify the type of the requirement according to the matched boilerplate

Requirements patterns are created by using a five-step wizard, the steps are the following:

- Step #1: provide an example to avoid add slots one by one
- Step #2: manage the syntax elements (slots) of the pattern
- Step #3: formalization as relationships (semantic graph)
- Step #4: formalization as meta-properties
- Step #5: administrative information

2.3.6 Suggestion management

The final responsibility on the ontology relies on domain experts. Nevertheless, it may happen that the requirements authors could identify new concepts to be added to the ontology, or new ways to express a requirement not matching with any of the agreed upon requirements patterns.

To sort out that problem, a suggestion mechanism has been created between RQA and RAT on the one side, and kM on the other side so that all the roles involved in the RBE process could participate in the evolution of the ontology.

Both, inference rules and thesaurus management sections of knowledgeMANAGER provide access to the suggestion mechanism. This allows the domain expert to accept or decline the suggestions sent by the requirements authors/reviewers; either, accepted or rejected, the suggestions remains in the system for further access, mainly from the engineer who issued the suggestion.

According to this schema, it's clear that a new profile is needed in the organizations to deal with the management of the ontology (not only based on suggestions). This picture depicts the roles envisaged for this RBE.

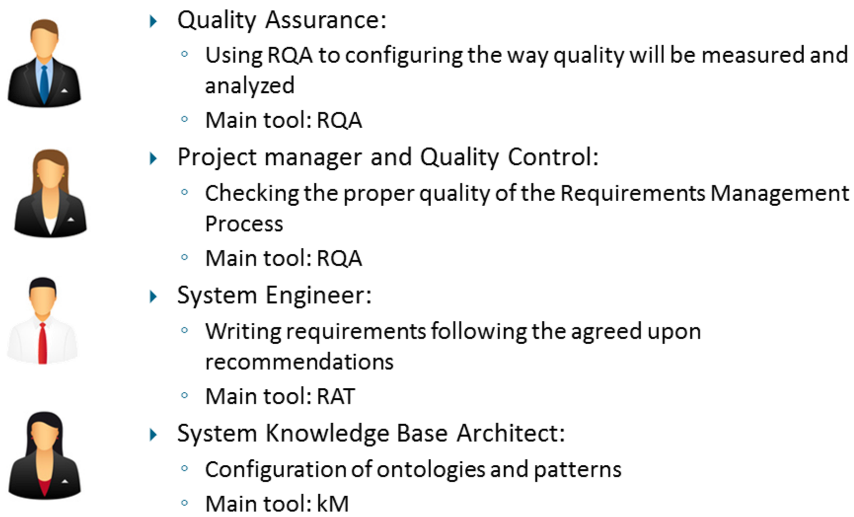

Figure 2-11. Roles for RBE

3 Training offered to partners

During the first few months of the project, several training sessions have been scheduled to:

- Train industrial partners on how to use RQA
- Train other technical partners involved in WP6.7 on the fundamentals and details about the semantic approach followed by RQS

The training material used for both sessions is available in the CRYSTAL repository:

- Training for end users (14 October 2013): https://projects.avl.com/11/0154/Data%20Exchange/Forms/AllItems.aspx?RootFolder=%2f11%2f0154%2fData%20Exchange%2f001_MEETINGS%2f011_SP6_Meetings%2fWP6_7%2fMeetings%2f2013-10-14%20RQS%20Training&FolderCTID=&View=%7bA036B3F1-CA9C-4631-A46F-C55BDA6D5C01%7d
- Training for technical partners (16 September 2013): https://projects.avl.com/11/0154/Data%20Exchange/Forms/AllItems.aspx?RootFolder=%2f11%2f0154%2fData%20Exchange%2f001_MEETINGS%2f011_SP6_Meetings%2fWP6_7%2fMeetings%2f2013-09-16_RBE%20Training%20about%20the%20tool%20bricks%20%28Madrid%29%2fkM%20Documentation&FolderCTID=&View=%7bA036B3F1-CA9C-4631-A46F-C55BDA6D5C01%7d

4 Main development goals during the CRYSTAL project

The set of features presented so far have been described as very valuable for the industrial partners related to WP6.7; nevertheless, all the partners involved in that workpackage are working on envisaging a set of new features to improve the tool.

The main improvements to RQS in general and kM in particular are the following:

- Integration in the overall tool chain through the IOS
- Import/export of ontologies
- Smart merging of ontologies
- Support to PBS (Product Breakdown Structure): by extending a particular type of *whole-part* relationships
- Provide collaborative management of ontologies along the supply chain
- Support to a set of new techniques for completeness checking between a requirements specification and the ontology
- Support to new correctness metrics:
    - Deprecated concepts
    - Use of not preferred concepts (synonyms)
    - Use of concepts identifies as ambiguous because of their list of more specific concept in the ontology
- Customized metrics: allowing the end-user to write the code for their own metrics
- Enhance collaborative work with RAT
- Enhance the current in and out-links metric with nominal links where the user could identify the name and direction of the link to quantify
- Pre and Post-analysis code: this represents a way for the end-users to write customized code to be executed at different particular moments during the quality analysis:
    - Before the analysis: the code will be able to change any of the attributes of the requirement
    - After the analysis: RQA will provide information related to the result of the analysis so that the proper actions could be taken
- Identification of similar requirements (through their semantic graphs) in previous projects
- Database with Requirements patterns for formal requirements, contracts, and meta-model concepts

5 Terms, Abbreviations and Definitions

CO	Confidential, only for members of the consortium (including the JU).
CRYSTAL	**CR**itical S**YST**em Engineering **A**cce**L**eration
D	Demonstrator
kM	knowledgeMANAGER
NL	Natural Language
O	Other
P	Prototype
PBS	Product Breakdown Structure
PP	Restricted to other program participants (including the JU).
PU	Public
R	Report
RAT	Requirements Authoring Tool
RBE	Requirements Based Engineering
RE	Restricted to a group specified by the consortium (including the JU).
RMS	Requirements Management System
RQA	Requirements Quality Analyzer
RQS	Requirements Quality Suite
SP	Subproject
UC	Use case
WP	Work Package

Table 5-1: Terms, Abbreviations and Definitions

6 References

[Author, Year]	Authors; *Title*; Publication data (document reference)