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

Sagem Use Case RBE Process and Tool Chain Evaluation V1
D204.010

DOCUMENT INFORMATION

Project	CRYSTAL
Grant Agreement No.	ARTEMIS-2012-1-332830
Deliverable Title	Sagem use case RBE process and tool chain evaluation V1
Deliverable No.	D204.010
Dissemination Level	CO
Nature	R
Document Version	V1.0
Date	2014-01-29
Contact	Marc Malot
Organization	Sagem
Phone	+33 1 58 11 41 33
E-Mail	marc.malot@sagem.com

AUTHORS TABLE

Name	Company	E-Mail
Marc Malot	Sagem	marc.malot@sagem.com

CHANGE HISTORY

Version	Date	Reason for Change	Pages Affected
0.01	2013-12-18	First document layout	All
0.02	2014-01-16	Update from “THE REUSE COMPANY” review	All
1.0	2014-01-28	Update from Cassidian and EADS IW reviews	All

CONTENT

Content of Figures

Content of Tables

1 Introduction

1.1 Role of deliverable

The work package WP204 will address an enhanced Requirements Based Engineering (RBE) process by using ontology and formal languages concepts.

Three iterations are planned for this work package.

The first iteration from Month 1 to Month 9 fulfills the following objectives:

- To define RBE industrial needs
- To initiate a desired RBE process
- To evaluate roughly the current version of the tools provided by WP607
- To identify the gaps between the current status of the tools and the desired process and to plan the CRYSTAL WP204 and WP607 development.

The second iteration will aim at specifying (and evaluating) an improved RBE process based on the evaluation results and gap analysis of the first iteration.

The third iteration will allow optimizing the RBE tool chain by using evaluation feedback, adding new functionalities and new tool connections (according to IOS specification if possible).

This document aims at describing the activities of the first iteration. It will be focused on:

- the objective of the work-package WP2.4
- the scope of the systems engineering process to be enhanced through CRYSTAL tool chain (mostly Requirements Based Engineering (RBE))
- the industrial use case of Sagem - an electrical based Flight control System – used for evaluation purposes
- the preliminary results of the evaluation of the current version of the RBE tool chain

1.2 Relationship to other CRYSTAL Documents

The work package WP204 is connected to:

- Work package WP607 which develops the RBE tool chain to match WP204 industrial needs by providing and integrating ontology based requirement engineering and requirements formalization. This deliverable is linked to the document D_607.011 (Specification, Development and Assessment for Requirements based Engineering) and to the deliverables D_607.021 (Requirements Quality Analyzer tool - RQA), D_607.031 (Requirements Authoring Tool - RAT), D_607.041 (knowledgeManager tool - kM).
- Work package 6.01 which develops the interoperability standard. This deliverable contributes to the document D_601.2-V1 (Interoperability Specification)
- Work package WP208 which summarizes in a public use case the common aeronautical engineering methods (including RBE). The Engineering Methods defined in this document are synchronized as much as possible with the document D_208.010 (CRYSTAL aerospace use case description) in order to build some shared objectives for interoperability standardization.
- Work package WP209 which deals with aeronautical ontology domain.
- Work package WP203 (Cassidian use case) which focuses on similar topics and with which we define common industrial needs for RBE.

1.3 Relationship to other Projects

The concepts and ideas that are analyzed within previous and on-going projects will be considered here.

ARTEMIS project CESAR will be considered with reference to many topics:

- Process specification
- Requirements specification including specification Completeness/Correctness/Consistency (CCC) criteria
- Ontology methods
- Formal languages

ARTEMIS project MBAT might be considered for interoperability topics.

1.4 Structure of this document

This document is organized as follows:

- **Section 2** recalls the goals of the WP204 and provides a description of the industrial case used by Sagem to evaluate the output from CRYSTAL.
- **Section 3** presents the industrial needs defined for the RBE tool chain and process.
- **Section 4** describes the principles of the systems engineering process to be designed and evaluated within CRYSTAL project, and the engineering methods identified so far for WP204. These engineering methods will be the input for interoperability services definition.
- **Section 5** provides a description of the current implementation status of the case.
- **Section 6** reports the activities of the preliminary evaluation of tools available at the beginning of the project within the first iteration phase in order to be able to better specify the gap to be filled out within the CRYSTAL project.

2 Use Case Description

2.1 Work Package Objectives

The lack of requirements quality often leads to additional efforts, cost overrun and schedule drifts in downstream development activities. One of the means to improve requirements quality is to formalize requirements using boilerplates, domain ontologies and patterns in order to allow automatic analysis and test generation.
Boilerplates provide requirements templates which consist of fixed syntax elements and attributes. The primary benefit of using boilerplates is that they allow requirements to be captured in a consistent fashion. Domain ontologies provide assistance in filling the attributes of boilerplates. Based on domain ontologies requirements quality analysis (including assessment of CCC (Completeness, Consistency, Correctness) and redundancy) can be automated.
According to each industrial process, an additional requirements formalization step should be optionally available based on pattern requirement capture (a smooth transition from natural language to boilerplates to pattern have been defined and validated during CESAR project). A pattern provides additional semantic restrictions to a boilerplate (syntax restrictions).
The requirement patterns allow deeper requirements analysis of the Completeness, Consistency and Correctness of a set of requirements, of system architecture consistency, and allow generating automatically test cases for requirements.

That is why, within WP 204, the main objectives will be to define a process with integrated tools to enhance Requirements Engineering (including DOORS requirements, requirement ontology, SysML models, and altarica models which will be covered in the next iteration of the deliverable according to the progress of the project…) in order to

- **To share common vocabulary within the project/organization/supply chain**
- **To enhance the quality of the specification at each step of development process**
- **To help the reuse of requirements from previous project.**

2.2 Sagem Electrical Flight Control System Description

The selected use case represents a typical kind of critical application for flight control command of an aircraft. It is representative of the kind of products SAGEM is able to design on the basis of its customer requirements. Therefore, ensuring that a consistent, complete and high quality set of requirements properly shared between customers and SAGEM is provided as input to design teams is a pre-requisite to target a quality product fully meeting customer expectations.

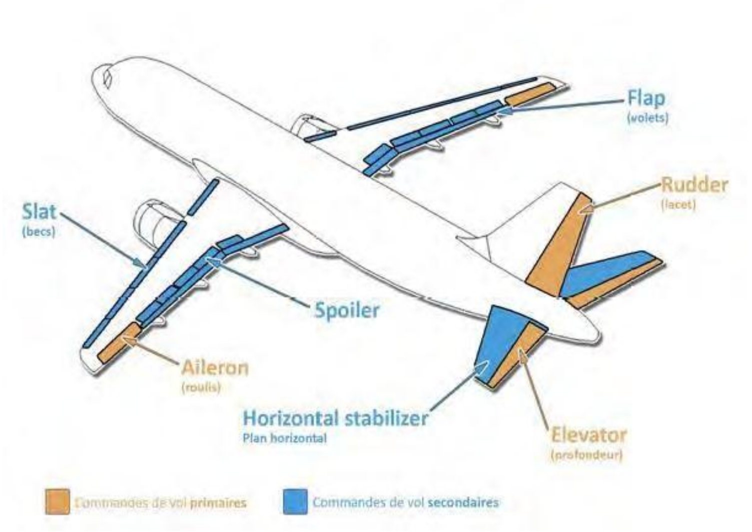


Figure 2-1: Sagem use case context presentation

First, the scope of WP204 use case will be limited to the “FLaps” Control System (FLCS) in order to keep the specification and modelling load low enough to focus on defining process, engineering methods and evaluating tool chain. Still the scope of the system is sufficient to test each of the objectives of the CRYSTAL project.

Secondly, the use case could be enriched to test if necessary the scale effects.

The scope of the FLCS is illustrated on following schema. The environment systems are in blue whereas the FLCS is in green. The behaviour is quite simple:

- The pilot use the Flap lever position to control the Flaps
- The Flight Control Computer receives measures of flaps positions through dedicated sensors. It calculates a flaps speed command. It also monitors the system.
- The FLCS receives the speed command to control the actuator and move the flaps.

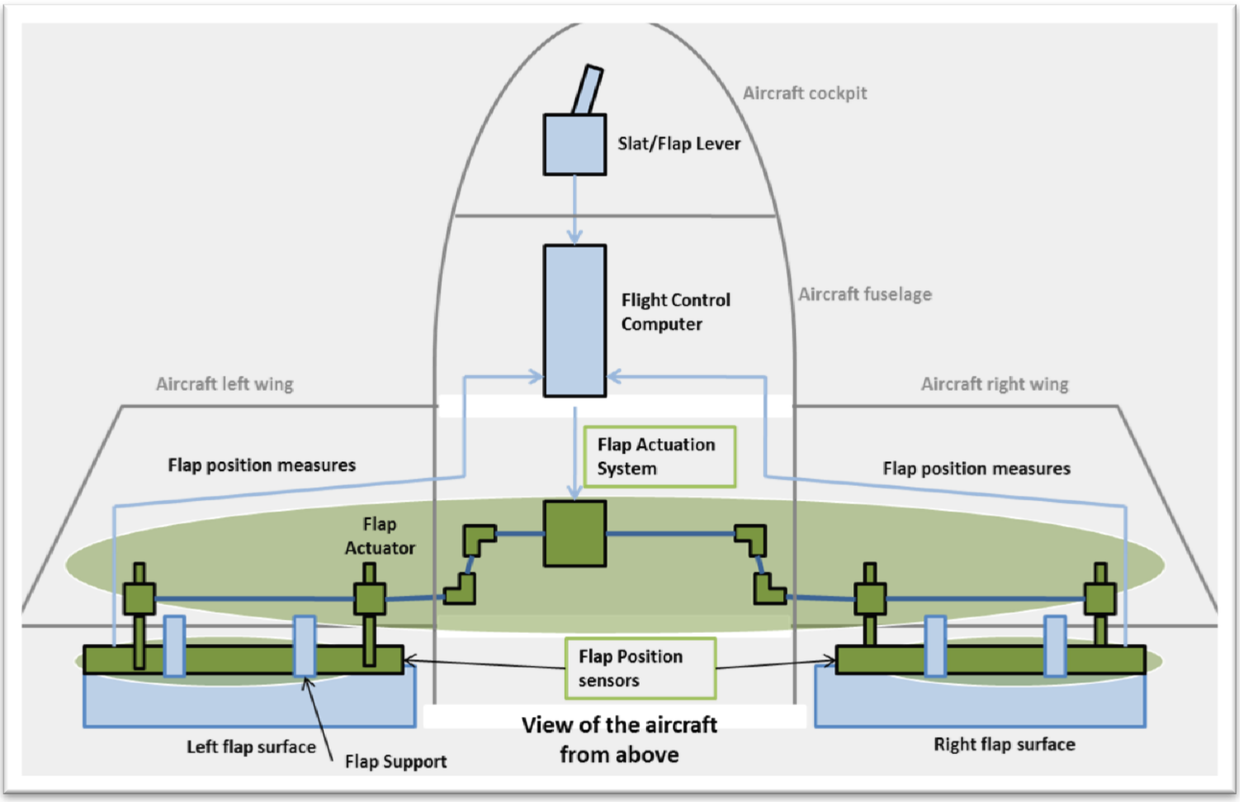


Figure 2-2: Scope of the Sagem use case: the Flaps Control System

The experience of Sagem in Flaps design will allow reusing some mature knowledge and requirements as a base to define the ontology to help specifying the Flaps Control System.

3 WP204 Industrial Needs for Requirements Based Engineering

3.1 Identification of Industrial Needs

Industrial needs for RBE process have been identified and discussed mainly with Cassidian, Alenia for industrial partners, The Reuse Company and OFFIS for solution providers.

The objectives in red font color constitute the “top objectives” which are:

- To provide means to help any participant to a project (or any stakeholder) all along the supply chain to share a common vocabulary in order to avoid errors due to misleadings. It is desired that this validated and centralized vocabulary be used in each artifact of the project (requirements, system model, safety model…).
- To improve the quality of the system specification using appropriate vocabulary and non-ambiguous syntactic sentences (defined wand suggested by validated boilerplates or patterns), taking benefit from knowledge stored in ontologies adapted to the system to be designed. The Completeness/consistency/correctness criteria defined during CESAR project can be used as a reference to measure the quality of a set of requirements and to provide some automatic checks.
- To support the reuse of (already validated) data produced in similar previous project (These data will be textual requirements in the first part of the CRYSTAL project, but could also be part of system models).
- To Integrate requirement tool chain through interoperability technologies based on IOS as much as possible
- To achieve a Technical Readiness Level of 6 for the requirements engineering process and the associated tool chain (meaning that prototype has been tested in a relevant environment)

Objectives for RBE tool chain	**Additional Information**
To share a common understanding	**Ontology Management**
Manage ontology libraries: Capability to define several ontologies and to add/merge them for a project	Basic idea is to have a modular approach that allows to re-use ontologies for different projects and to share ontologies between OEM and supplier. Universal physical quantities ontology can be used in almost every project. A base aerospace ontology could provide concepts like pilot or aircraft. For specialist domains like safety, dedicated ontologies can be developed.
Merge ontologies: Check of non-inconsistency of these associated ontologies	Ontology-based quality checks shall be performed during merge (import) operations. One important check is to detect conflicts after merging two ontologies.
Use of the most adapted terms: Definition of standard/normed words validated by experts. - Relation between standard word and currently used terms	The idea is to propose to the requirement engineer some normed/standard terms already validated by experts as soon as an associated (link to be defined in the ontology) word is used in a sentence. For instance: the checker can propose to replace "the aircraft" by "A380" or "A350" and so on... if the ontology is customized to encourage the use of a specific name of aircraft because the aircraft term is considered to be not precise enough.
Evaluate ontologies obsolescence: Calculate terminology frequency in specifications	Besides quality checks, it can be interesting to see how many instances of the concepts exist in the requirements specification. This allows identifying concepts which are less relevant or obsolete.

Objectives for RBE tool chain	**Additional Information**
Ontology Evolution based on Terminology frequency calculations:	Besides quality checks, it can be interesting to see how many instances of the concepts exist in the requirements specification. This allows measuring the completeness of ontology by the usage of concepts in requirements specifications.
Exchange ontologies: Capability to share data along the supply chain -To share wording, definition, relations between concepts	A standard format and ontology retrieval services allow to exchange (deploy) ontologies between different applications (e.g. analyze requirements, check architecture) and along the supply chain.
Extract ontologies: capability to learn ontologies from scratch or to improve existing ontologies	Existing specifications or glossaries may be analyzed to support ontology extraction (learning) in a semi-automatic way.
To improve the quality of requirements	**Boilerplate Management**
Capability to define some homemade boilerplates	Define an extensible library of boilerplates.
Provide boilerplate groups: Capability to provide some on purpose validated boilerplates according to the topics	Group boilerplates according to the level (e.g. stakeholder, system...), domain (safety, environmental, hardware…) and type (functional, performance, interfaces…). Some of these should be addressed to the certification mapping
Capability to support formalization of requirements	The structure of boilerplates should be exploited to support the transition to formal specifications (e.g. patterns, contracts). Additional CCC checks can be performed using formal methods.
	Requirements Quality Assessment
Create metrics: Capability to create homemade metrics	Extend the standard metrics by new metrics. Metrics may be composed of other metrics. Validity of metrics definition shall be checked. Metrics definitions may be organized in libraries and shared between projects.
Detect weak phrases: RAT should guide the writer to use non ambiguous words, standards words or words assessed by experts. A list of validated words related to the current natural language should be proposed by RAT when links exist in ontology.	It should be possible to maintain a list of weak phrases (ambiguous words) which are checked automatically. It should also be possible to define some synonyms, or nearly synonyms…
Analyze requirements quality: Analyze of completeness/correctness/consistency criteria	Standard quality metrics such as CCC shall be supported. Refer to next section.
Identify requirement type	Define taxonomy of requirements types. Classify requirements according to the type taxonomy. This allows clustering requirements for analysis and verification. For example, provide all functional requirements to perform functional analysis or retrieve all process requirements to prepare a project audit, etc.

Objectives for RBE tool chain	Additional Information
To support the reuse strategy	**Requirements Reuse**
Capability to identify/suggest similar requirements within previous projects when writing requirements while respecting the potential limitation of the access right to the previous projects	However, reuse should not be limited to requirements. Reuse of architectural design, validation and verification is also beneficial. This can be achieved by exploiting traceability. An important link can be made to variability management (product family management). When similarities between different project requirements are identified, they may be extracted as common features and pro-actively managed in a product family engineering approach. In military domain, it could be forbidden to have access to some classified data. The access right to previous project requirements should be managed.
Capability to customize the research base	Selection of projects, deliverables, domains to be regarded.
Generic needs	**Administration**
Capability to collaborative work	Create and use ontologies, boilerplates and metrics in a collaborative environment. This may imply using a common shared repository, locking, conflict resolution, …
Capability to manage access rights -Different levels of privacy (inside ontologies) -Different levels of rights	Define permissions such as read, modify, create, and delete ontology libraries, boilerplates and metrics. Ideally, a role-based model is used for access rights (e.g. ontology engineer, requirements analyst, quality manager). In military domain, it could be forbidden to have access to some classified data. The access right to different project requirements should be managed.
Capability to manage the configuration	Tool versions need to be defined. Requirements under analysis need to be baselined together with ontology, boilerplates and analysis results. It should be possible to exactly reproduce the analysis results.
	Traceability
Modification tracks on requirements	Requirements changes need to be recorded (author, date) in the history. This needs to be considered if an improved requirement is sent back to DOORS. Changes against baselines need to be justified by a trace to a change request.
Traceability between artifacts	Traceability between requirements (on different levels), between requirements and design, between requirements and validation, between requirements and verification needs to be supported. Traceability to the engineering analysis and to its results to be supported.
Analyze requirements traceability: Analyze CCC criteria with respect to traceability links.	Analyze traceability links between requirements: identify potentially missing or incorrect links based on analysis.

Objectives for RBE tool chain	Additional Information
	Usability
Filter requirements under analysis	Reduce the scope of analysis, i.e. reduce non-requirements such as section heading. Potentially an attribute can be used to identify requirements. Provide capability to remove from the list requirements from the list which do not need to be re-checked.
Provide guidance	Provide guidance (documentation, tutorials, and examples) either built-in or separate from the tools that helps in proper design of ontologies and boilerplates.
	Tool Qualification
Provide tool qualification package	RQA can be used to automatically check some rules of a requirements standard. If we seek for certification credit, we need to fulfill the configuration control requirements and perform tool qualification. This is similar to static code analysis. RTCA/DO-330 (Software Tool Qualification Considerations) provides guidance on this topic.
	Customization
Capability to customize the integration with Users' DOORS process (requirements number, status of requirement validation, filter…)	This includes requirements identification, pre- and post-processing steps (e.g. changing attributes like the requirements status) when requirements are retrieved from DOORS and sent back to DOORS. The latter can be accomplished by triggering DXL scripts in DOORS.
	General
To Integrate requirement tool chain through interoperability technologies based on IOS as much as possible	IOS adapter need to be implemented. Requirements shall be exchanged based on IOS. The results of quality assessment shall be available via IOS. Project monitoring tools may use these results, too.
To raise the Technical Readiness Level of the RBE tool chain and process to 6	TRL 6: prototype system that is tested in a relevant environment.

Table 3-1: Industrial needs for WP204 process and tool chain

3.2 Specification of Requirements Quality Criteria

The quality criteria of a set of requirements have been studied during the ARTEMIS CESAR project and have been classified in the document [Malot/CESAR, 2012] (CESAR D_SP2_R3.3_M3_vol 4: Completeness/Consistency/Correctness).

These CCC criteria have been reused within the WP204 objectives as requirements quality to be improved through the developed RBE process.

The document [Malot/CESAR, 2012] contains for each of the criterion:

- its origin from one or several embedded world standards,
- the clarification of their meanings
- some instances to show their application and utility
- some CESAR proposals to improve the coverage of the criterion (i.e. to enhance the confidence that the criterion is respected by the specification)

These descriptions are not recalled inside the document because of the size of the CESAR document and because this document is public.

The criteria have been classified into three main categories:

- Completeness
- Consistency
- Correctness.

Completeness criteria according to CESAR project:

The figure below proposes an overall vision of the notion of completeness for requirements.
Two complementary approaches can be taken to study completeness:

- Make sure that all categories of requirements (by categories we understand environment requirements, functional requirements, maintenance requirements...) are covered. This is represented below by the use of viewpoints.
- The set of requirements is generated from two sources: the first one external to the system to be specified itself (stakeholder needs, higher level requirements), and the second one being all additional input brought to the system (assumptions based on lessons learnt, internal development constraints…)

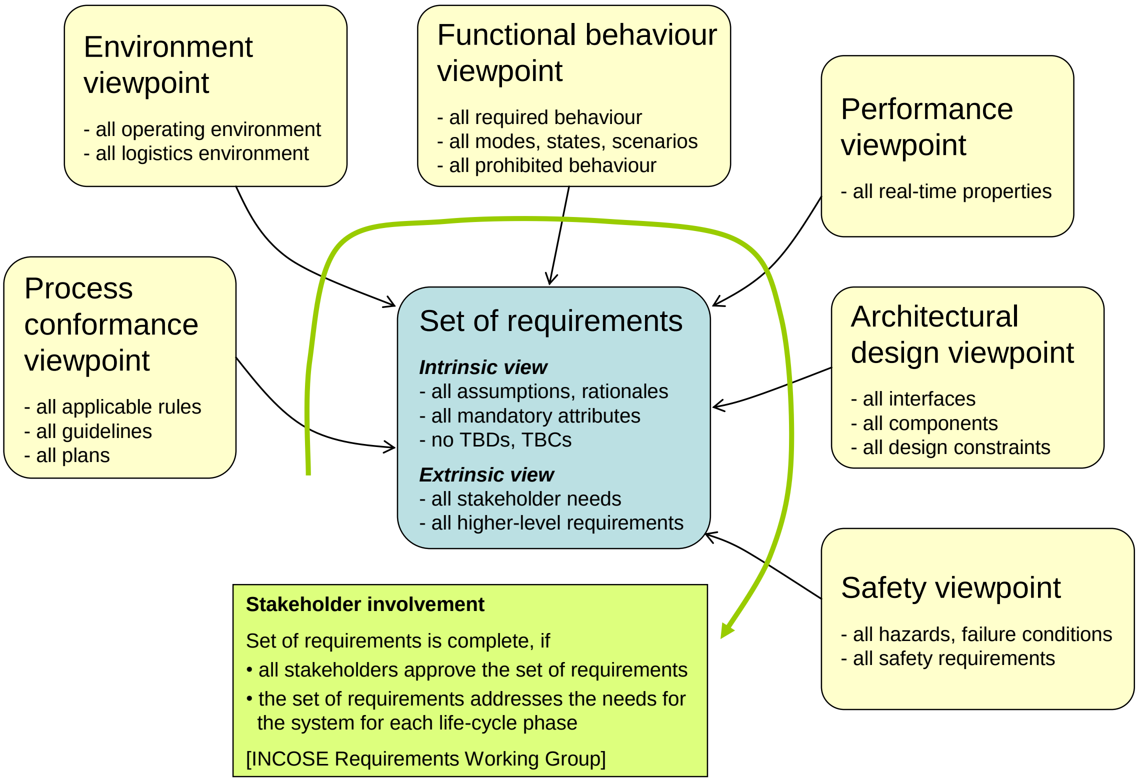


Figure 3-1: Overview of requirement completeness

Viewing this picture and seeing the word "*all*" in every bubble, identifies the problem: how can we guarantee that we did indeed identify all requirements for each of these categories?

Correctness criteria according to CESAR project:

The figure below proposes an overall vision of the refinement of the notion of correctness of requirements.

This classification can be considered as arbitrary and might thus not be the best suited. However, the main intent is to group the requirements of the several standards.

Each of the criteria identified is described in the following chapters.

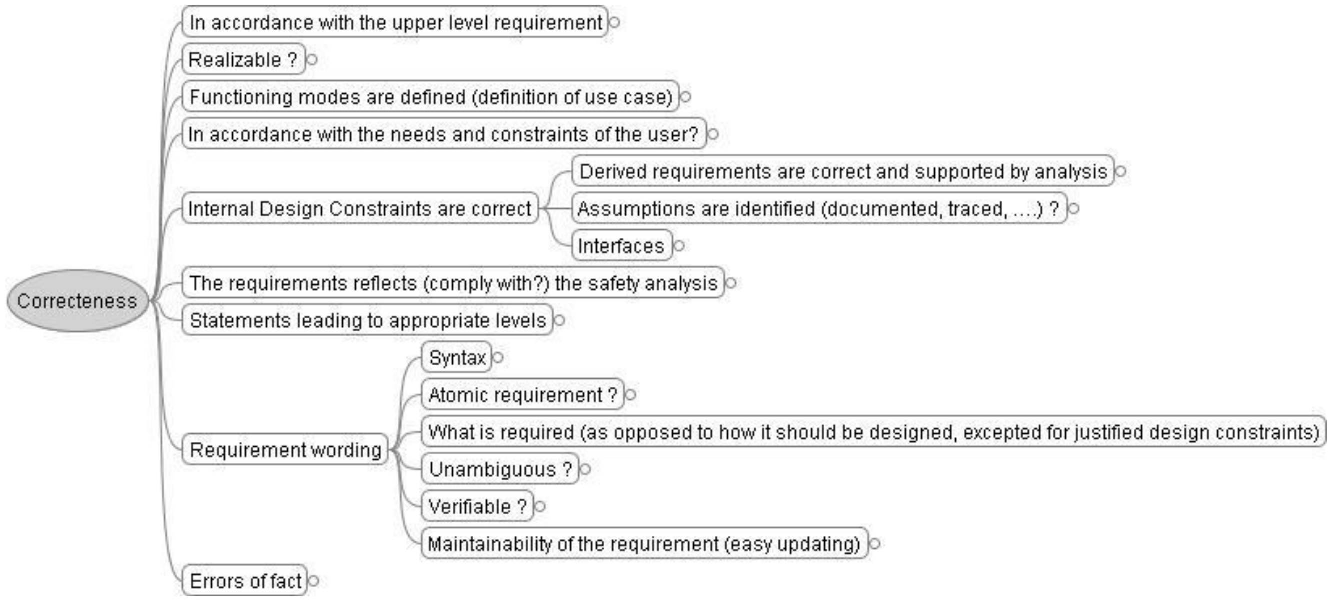


Figure 3-2: Overview of correctness criteria

Consistency criteria according to CESAR project:

The project CESAR also describes some criteria to check the consistency of a set of requirements. System engineers have for instance to verify if:

- Requirement does not conflict with other and with higher level requirements?
- Requirements are not redundant?

4 Identification of Engineering Methods

4.1 Systems Engineering Process Description

This section aims at defining the engineering methods Sagem would like to experiment within the CRYSTAL project on Flight Control System use case (probably not of all of these engineering methods will be used during CRYSTAL project but the choice will be done according to the progress of the work).

Some are specifics to Sagem use case, some are common with other use cases and have been basically defined in collaboration with WP2.08 aeronautical public use case.

The tool chain for the WP204 will contain the following elements:

- DOORS requirements
- Rhapsody SysML models
- Requirements Quality Suite (RAT, RQA, kM, OFFIS algorithms…)
- Altarica tool to address safety purpose and more generally models consistency topics (this tool will be confirmed in the next iteration of this deliverable according to the progress on other topics).

The figure 2-1 illustrates this tool chain which should be connected through IOS and the relation desired with the supply chain.

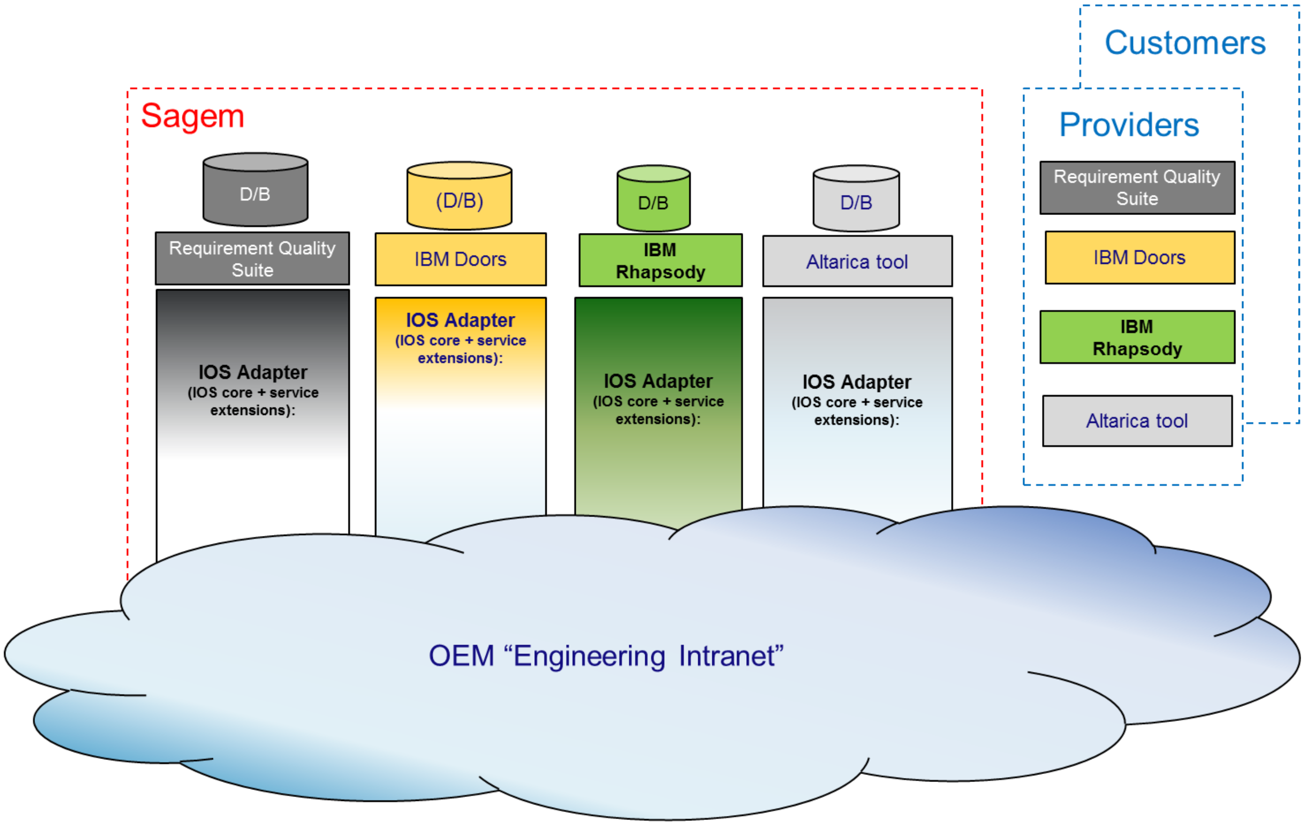


Figure 4-1: WP204 preliminary tool chain

The main engineering methods Sagem is focused on are:

- Method “Support analyzing and reusing Requirements” (by using ontologies and knowledge databases)
- Method “Support analyzing and reusing SysML Models” (by using ontologies and knowledge databases)
- Method “Verify consistency/completeness/correctness between requirements and models”
- Method “Define ontology from existing artefacts” (To be defined in next iteration of this deliverable)
- Method “Fault-tree generation”
- Method “Verify design against requirements”
- Method “Trade-off analysis”
- Method “change Impact analysis”
- Method “Maintain consistency between multi-viewpoint models”
- Method “Search Data”
- Method “Provide specification”
- Method “Put all data under configuration control”

4.2 Specific methods for WP204

4.2.1 Method “Support analyzing and reusing Requirements”

The general purpose of this engineering method is to improve the quality of requirements defined in a textual format. It is assumed that at least two tools are involved, one tool for managing the textual requirements under configuration, and another tool for analyzing the quality of the textual requirement. The interoperability need in this scenario therefore concerns the interaction between the requirements management and the requirements quality analysis tool.

<table>
<tr><th colspan="3">Engineering Method: UC204_SupportAnalyzingAndReusingRequirement_001</th></tr>
<tr><td colspan="3">Purpose: The Requirement Engineer wants to capture, reuse former requirement if possible, and then check the quality of a requirement using RQS</td></tr>
<tr><td colspan="3">Comments:</td></tr>
<tr><th>Pre-Condition</th><th>Engineering Activities
(made of steps)</th><th>Post-Condition</th></tr>
<tr><td>Requirements from previous projects are stored in Doors D/B

Requirement quality tools customization has been defined in RQS and is stored in D/B (ontology, terms, boilerplates, patterns, metrics are defined and customized for the project)</td><td>1 – initialize the project
1.a. RQS will send the “Identify the available project modules according to my credentials” action to start assessing the quality
1.b. DOORS sends back the available projects or folders with some modules inside list
1.c. in RQS, we select a DOORS project or folder to import (Requirements for the project, but also requirements from previous projects to allow the reuse of requirements)
1.d. RQS sends to DOORS a “get modules for project or folder at the first level or at any level” action.
1.e. DOORS sends to RQS the list of modules inside that project or folder along the module attributes
1.f. RQS sends to DOORS a “get requirements for a set of modules” action
1.g. DOORS sends to RQS the requirements stored in the desired modules along with their attributes, values, their historical changes and links (link module name, in or out direction, and target/source requirement id and module id) all of them with the reference to the module where they are stored.
1.h. RQS can ask for the module properties and full name and path of a single module

2 - Specify/write a requirement
2.a RQS will send a “add a requirement” action to DOORS D/B
2.b. DOORS will report success or failure for the creation action
2.c. RQS will send a “update a requirement “ action to DOORS
2.d. DOORS will report success or failure for the update action2.e. RQS will send a “remove a requirement” to DOORS
2.f. DOORS will report success or failure for the update action
2.g. RQS will send a “update and save a requirement including attributes” action
2.h. DOORS will report success or failure for the update action
2.i. in RQS, we use “get one single requirement along with their attributes and links” action when we have tried to save a requirement and that this action have failed because someone else was working in same time in collaborative mode and had simultaneously updated this requirement. “get one single requirement along with their attributes and links” service will require this updated requirement.</td><td>Requirements for the project have been written, quality checked, approved or modified.</td></tr>
</table>

	3- Reuse requirements from previous projects 3.a RQS will send a “get the possible link modules for a pair or DOORS modules” 3.b DOORS will return all the available link module identifies for those modules 3.c. RQS will send a “add a link to a requirement in a link module” 3.c. DOORS will report success or failure for the update action 3. RQS will send a “get one single requirement from a previous project” action 3.f. DOORS will return the desired requirement along with their attributes, values, their historical changes and links (link module name, in or out direction, and target/source requirement id and module id) **4- Check the quality of a set of requirements** 4.a. RQS will send a “add an attribute of a set type in a DOORS module” action 4.b. DOORS will report success or failure for the adding action 4.c. RQS will send a “remove an attribute from a DOORS module” action 4.d. DOORS will report success or failure for the deletion action4.e.RQS will send a “check if an attribute exists in a DOORS module” action 4.f. DOORS will report whether the attribute exists or not 4.g. RQS will send a “check if an attribute with a set type exists in a DOORS module” action.4.h. DOORS will report whether the attribute exists or not 4.i.RQS will send a “get the type of an attribute in a DOORS module” action 4.j. DOORS will give back the attribute type of the attribute 4.k. RQS will send a “get the attribute value of a requirement in DOORS” action 4.l. DOORS will give back the value of the attribute 4.m. RQS will send a “get all the attributes and values of a requirement in DOORS” action. 4.n. DOORS will give back the name and value of all attributes of the requirement 4.o. RQS will send an “update action of any attribute value of a requirement in DOORS” 4.p. DOORS will report success or failure for the update action	

Artefacts Required as inputs of the Activities		Artefacts used internally within the Activities (optional)		Artefacts Provided as outputs of the Activities	
Name	Requirement	Name	DOORS internal Requirement	Name	Requirement
Generic Type: (Tool or language independent type)	Natural Language Requirement	Type:	Requirement as stored in DOORS D/B	Generic Type: (Tool or language independent type)	Natural Language Requirement
Required Properties: (Information required in interactions between steps)	**DOORS D/B::** **Project/Folder:** -Path -Name -Identifier -Modules at the first level **Module:** - Module identifier - Module location - All other attributes - Requirements inside the module **Requirement:** -Requirement ID - Requirement Statement in natural language - Requirement Version - Links - Version history - All other attributes **Link modules:** -Link name -Path	Properties:	**DOORS D/B:** **Project/Folder:** -Path -Name -Identifier -Modules at the first level **Module:** - Module identifier - Module location - All other attributes - Requirements inside the module **Requirement:** -Requirement ID - Requirement Statement in natural language - Requirement Version - Links - Version history - All other attributes **Link modules:** -Link name -Path	Provided Properties: (Information provided in interactions between steps)	**DOORS D/B:** **Requirement:** - Requirement ID - Requirement Statement in natural language - Requirement Version - Reviewing state (e.g., "checked", "unchecked" state to be set by the RQA Requirement Engineer) - Requirement Type (e.g., simply encoded in a string, "weight", "safety", "maintainability", "functional", etc.) - Changes in other attributes - Quality Level - Quality Summary - Quality Date - Quality Value
Description & Interoperability Additional constraints: For the first steps of the activity, basic properties are required by the Requirement Engineer in order to pick up the ones he/she is interesting in analyzing.		Description: RQS will analyze the requirement quality and structure to match the company writing rules		Description & Interoperability Additional constraints:	
		Name	RQA internal Requirement		
		Type:	Formal representation of the requirement are stored in RQS Quality D/B		
		Properties:	TBD		

4.2.2 Method “Support analyzing and reusing SysML Models”

The general purpose of this engineering method is to improve the quality of models defined in a SysML format. The notion of quality of models has to be defined during next iteration of the use case.

It is assumed that at least two tools are involved, one tool for managing the SysML models under configuration (IBM Rhapsody), and another tool for analyzing the quality of the models. The interoperability need in this scenario therefore concerns the interaction between the models management and the models quality analysis tools. This interaction has to be confirmed in next iteration of the use case.

The operation will be the analogous to the method “Analyze and reuse requirements”.

4.2.3 Method “Verify consistency/completeness/correctness between requirements and models”

This engineering method is related to consistency checks methods defined for WP208. It would be addressed the benefit of the ontology to help guaranteeing consistency/completeness/correctness between SysML model and requirements. These checks have to be elaborated in more details for the third iteration of the use case.

4.3 Overview of methods reused from WP2.08

The following engineering methods are common with WP2.08 and are described in Annex I:

- Method “Fault-tree generation”
- Method “Verify design against requirements”
- Method “Trade-off analysis”
- Method “change Impact analysis”
- Method “Maintain consistency between multi-viewpoint models”
- Method “Search Data”
- Method “Provide specification”
- Method “Put all data under configuration control”

5 Use Case Implementation Status

5.1 Tools installation

The following tools have been installed in order to begin evaluation of WP204 tool chain.

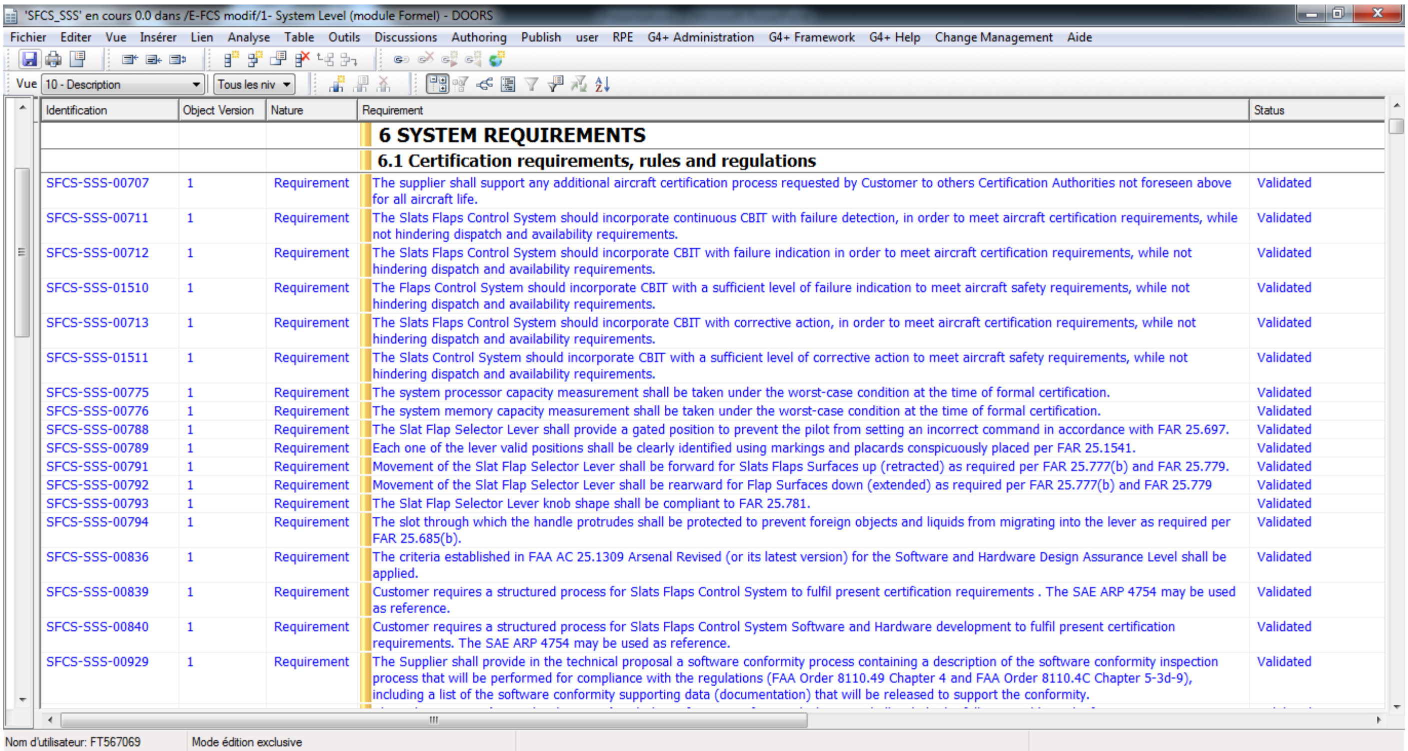
- IBM DOORS 9.3
- IBM Rationale Rhapsody 7.6 (might be updated later to Rhapsody 8.03)
- The Reuse Company Requirements Quality tools
  - Requirement Authoring Tool (RAT issue 4.1)
  - Requirement Quality Analyzer (RQA issue 4.1.4892.24110)
  - Knowledge Manager (kM issue 6.1.4892.23775).

Two more iterations of Requirements Quality tools including the OFFIS analysis should be analyzed during CRYSTAL project.

A safety tool might be added according to the progress on the tool chain.

5.2 Requirements Implementation

A System Specification of the FLCS has been defined in DOORS.



'SFCS_SSS' en cours 0.0 dans /E-FCS modif/1- System Level (module Formel) - DOORS

Fichier Editer Vue Insérer Lien Analyse Table Outils Discussions Authoring Publish user RPE G4+ Administration G4+ Framework G4+ Help Change Management Aide

Vue 10 - Description Tous les niv

Identification	Object Version	Nature	Requirement	Status
			6 SYSTEM REQUIREMENTS	
			6.1 Certification requirements, rules and regulations	
SFCS-SSS-00707	1	Requirement	The supplier shall support any additional aircraft certification process requested by Customer to others Certification Authorities not foreseen above for all aircraft life.	Validated
SFCS-SSS-00711	1	Requirement	The Slats Flaps Control System should incorporate continuous CBIT with failure detection, in order to meet aircraft certification requirements, while not hindering dispatch and availability requirements.	Validated
SFCS-SSS-00712	1	Requirement	The Slats Flaps Control System should incorporate CBIT with failure indication in order to meet aircraft certification requirements, while not hindering dispatch and availability requirements.	Validated
SFCS-SSS-01510	1	Requirement	The Flaps Control System should incorporate CBIT with a sufficient level of failure indication to meet aircraft safety requirements, while not hindering dispatch and availability requirements.	Validated
SFCS-SSS-00713	1	Requirement	The Slats Flaps Control System should incorporate CBIT with corrective action, in order to meet aircraft certification requirements, while not hindering dispatch and availability requirements.	Validated
SFCS-SSS-01511	1	Requirement	The Slats Control System should incorporate CBIT with a sufficient level of corrective action to meet aircraft safety requirements, while not hindering dispatch and availability requirements.	Validated
SFCS-SSS-00775	1	Requirement	The system processor capacity measurement shall be taken under the worst-case condition at the time of formal certification.	Validated
SFCS-SSS-00776	1	Requirement	The system memory capacity measurement shall be taken under the worst-case condition at the time of formal certification.	Validated
SFCS-SSS-00788	1	Requirement	The Slat Flap Selector Lever shall provide a gated position to prevent the pilot from setting an incorrect command in accordance with FAR 25.697.	Validated
SFCS-SSS-00789	1	Requirement	Each one of the lever valid positions shall be clearly identified using markings and placards conspicuously placed per FAR 25.1541.	Validated
SFCS-SSS-00791	1	Requirement	Movement of the Slat Flap Selector Lever shall be forward for Slats Flaps Surfaces up (retracted) as required per FAR 25.777(b) and FAR 25.779.	Validated
SFCS-SSS-00792	1	Requirement	Movement of the Slat Flap Selector Lever shall be rearward for Flap Surfaces down (extended) as required per FAR 25.777(b) and FAR 25.779	Validated
SFCS-SSS-00793	1	Requirement	The Slat Flap Selector Lever knob shape shall be compliant to FAR 25.781.	Validated
SFCS-SSS-00794	1	Requirement	The slot through which the handle protrudes shall be protected to prevent foreign objects and liquids from migrating into the lever as required per FAR 25.685(b).	Validated
SFCS-SSS-00836	1	Requirement	The criteria established in FAA AC 25.1309 Arsenal Revised (or its latest version) for the Software and Hardware Design Assurance Level shall be applied.	Validated
SFCS-SSS-00839	1	Requirement	Customer requires a structured process for Slats Flaps Control System to fulfil present certification requirements . The SAE ARP 4754 may be used as reference.	Validated
SFCS-SSS-00840	1	Requirement	Customer requires a structured process for Slats Flaps Control System Software and Hardware development to fulfil present certification requirements. The SAE ARP 4754 may be used as reference.	Validated
SFCS-SSS-00929	1	Requirement	The Supplier shall provide in the technical proposal a software conformity process containing a description of the software conformity inspection process that will be performed for compliance with the regulations (FAA Order 8110.49 Chapter 4 and FAA Order 8110.4C Chapter 5-3d-9), including a list of the software conformity supporting data (documentation) that will be released to support the conformity.	Validated

Nom d'utilisateur: FT567069 Mode édition exclusive

Figure 5-1: FLCS DOORS requirements

5.3 Modelling Implementation

A SysML model of the FLCS is currently being created. This model aims at supporting the process to specify the system. That is why this model is planned to be connected to DOORS specification and if possible in a second time with ontology tools.

Hereafter is described a SysML Block Definition Diagram which allows defining the external actors and system in relation with the FLCS.

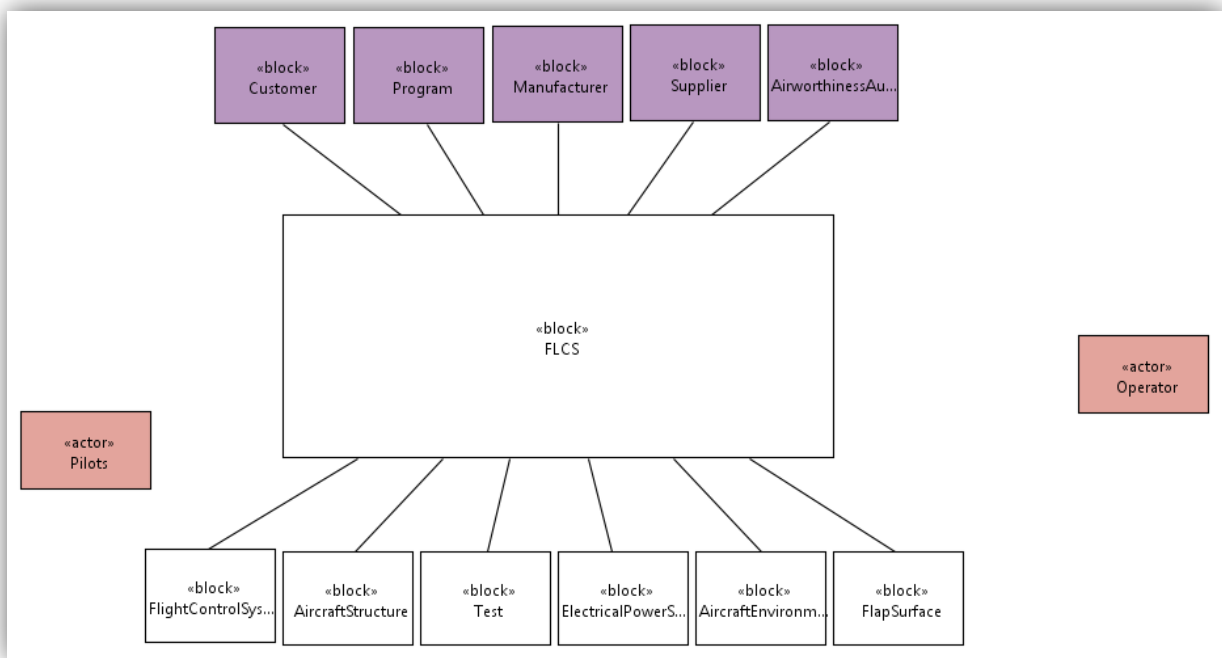


Figure 5-2: Context Block Definition Diagram

The following figure presents a statechart defining the lifecycle of the FLCS. This diagram allows with the former to communicate with customers to validate that all the stakeholders and all the use conditions have been considered for system specification.

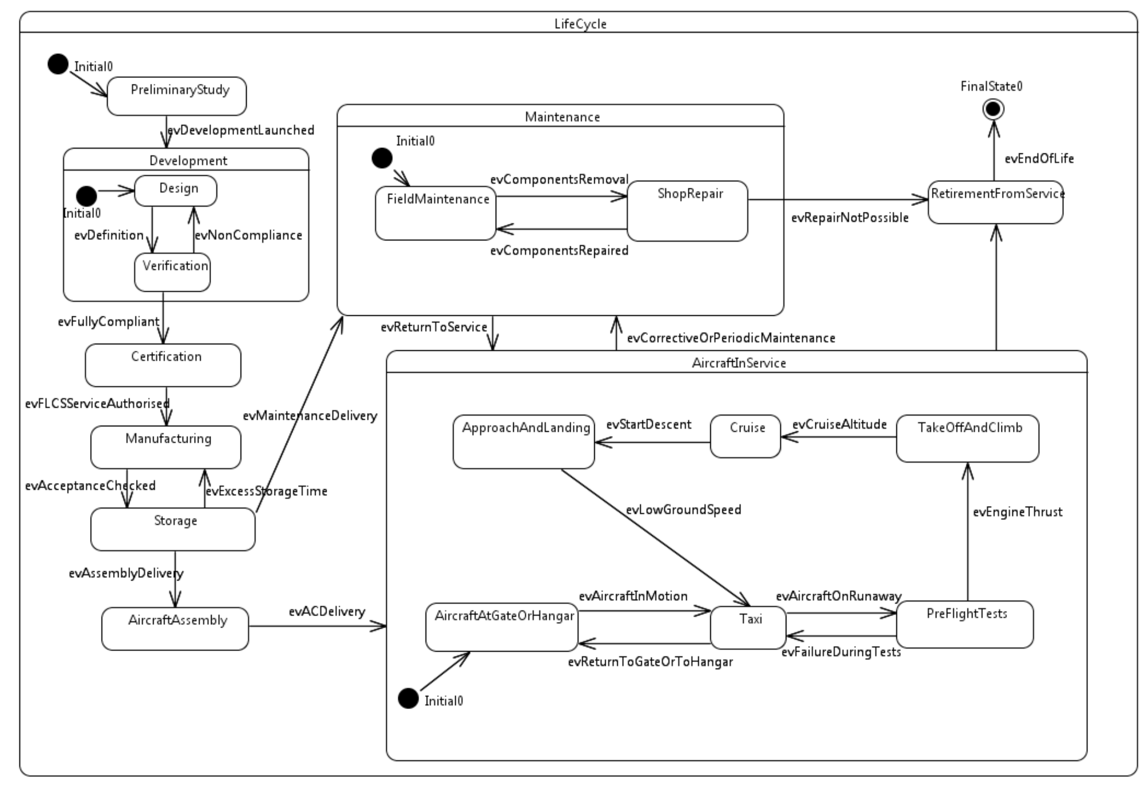


Figure 5-3: Lifecycle Statechart Diagram

6 Preliminary Tool Chain Evaluation

This section aims at presenting the results of the preliminary evaluation of tools available at the beginning of the project within the first WP204 iteration phase.

These results are used to identify the gap to be filled within the CRYSTAL project between industrial users' needs and current version of the tools. They allow consolidating a roadmap and to better specify/refine specification for the second iteration phase.

As OFFIS was part of the CESAR project, Sagem was able to have a first overview of their consistency check algorithms. That is why within first iteration phase, the evaluation effort of Sagem was focusing on the "TRC" tools (knowledgeManager, Requirements Authoring Tool, Requirement Quality Analyzer) to acquire the concepts of these tools, to verify their alignment with Sagem objectives.

These tools are described in details in deliverables D607.021 (Requirements Quality Analyzer tool - RQA), D607.031 (Requirements Authoring Tool - RAT), D607.041 (knowledgeManager tool - kM).

6.1 First steps with the Current Tools

6.1.1 First steps with "knowledge Manager" tool

An ontology was created by Sagem in order to be able to evaluate RAT and RQA capabilities. Some semantics, some thesaurus (a graphical instance is given figure 6-1), several sentence patterns (safety topics, environment standard conformance) have been defined.

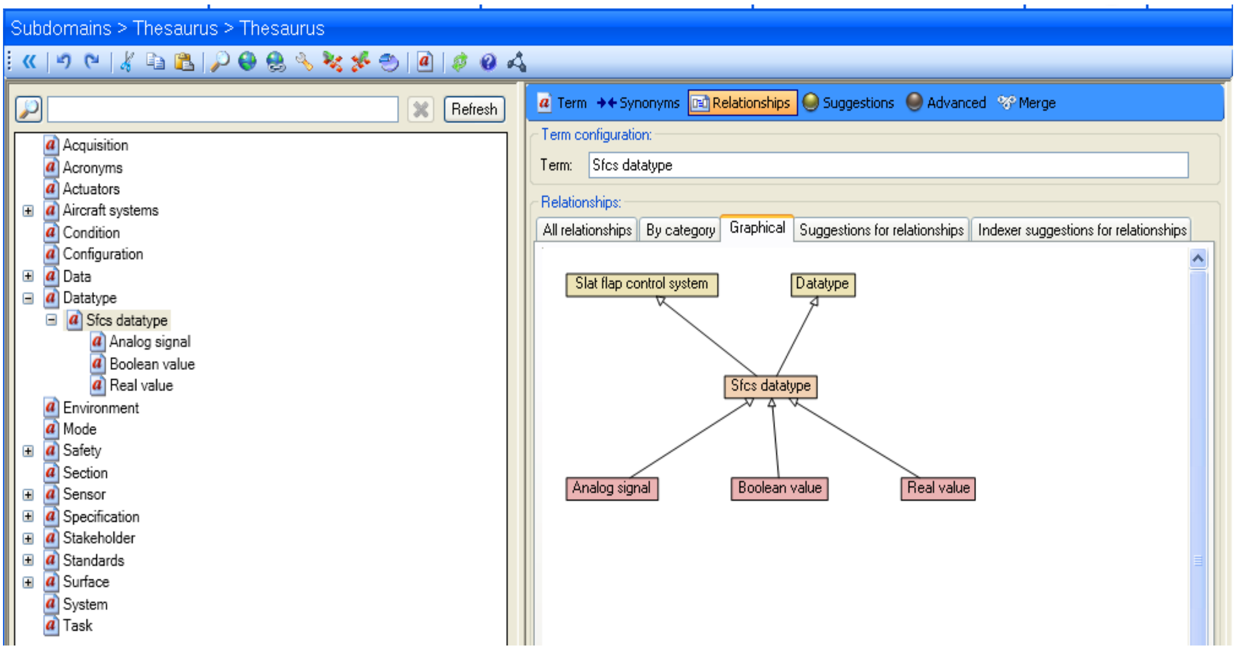


Figure 6-1: Graphical representation of FLCS thesaurus (knowledgeManager tool screenshot)

We succeeded in using this tool after a quite long training period. As this tool will be used by internal ontology managers (a few number of specialists), this could be acceptable but we think that the ergonomic of the tool should be enhanced and that some low level operations currently needed to define patterns should be integrated directly in the tool (tokenization etc.).

Note: The first version of the ontology was not structured to enable reuse on different project so far. This is an objective for the next iteration.

6.1.2 First steps with "RQA" tool

The RQA tool uses the knowledge stored in knowledgeManager tool (metrics, patterns, terms…) to evaluate the quality of a specification.
Each requirement is classified for each metric within low/medium/high quality categories.

The figure 6-2 illustrates some of the metric already available and customizable in RQA tool. These metrics will be completed to cover as much as possible the CCC criteria.

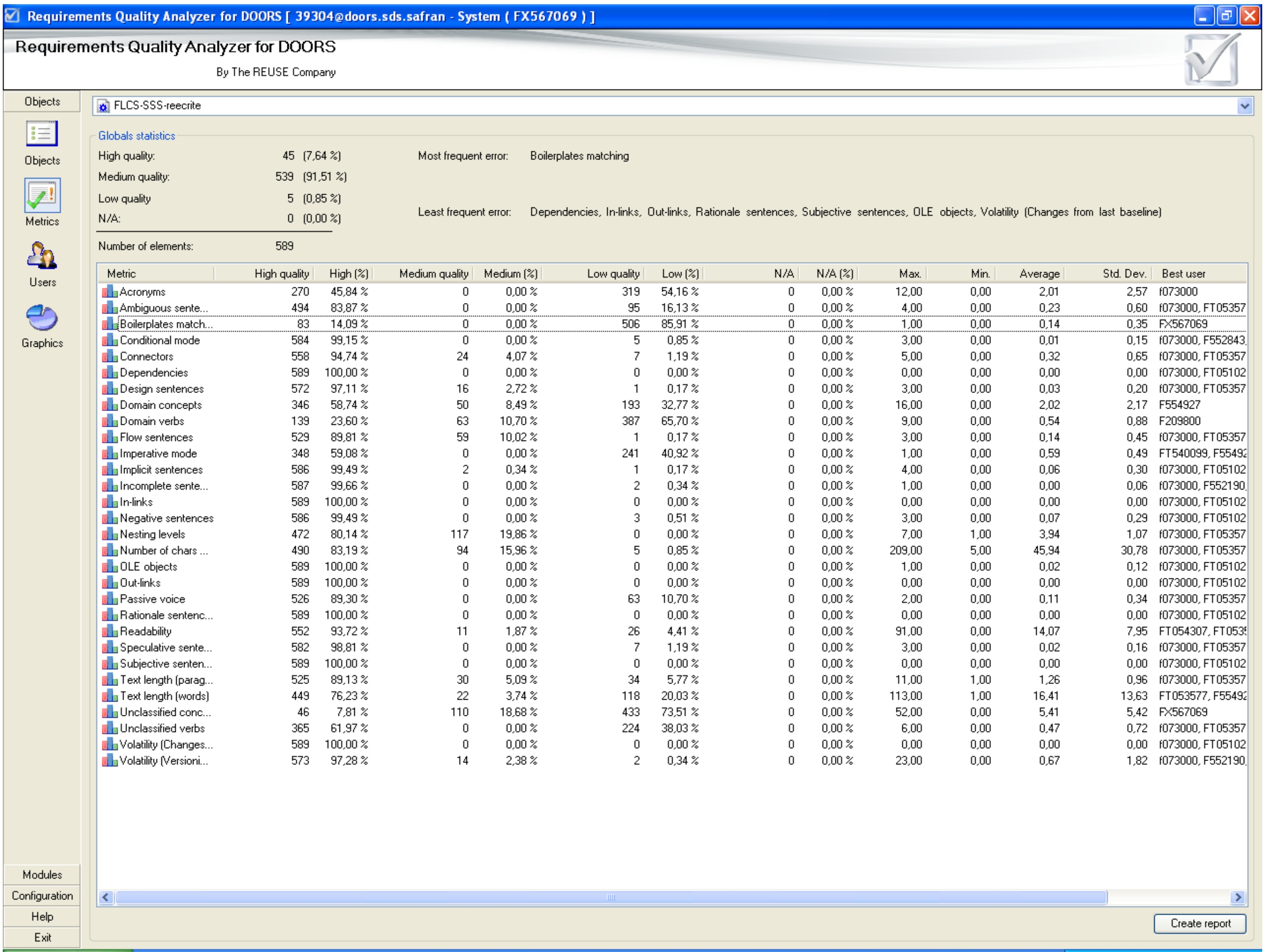
Sagem have not worked on metric customization so far.

Metric	Description	Default content
Acronyms	Avoid using acronyms which are not declared into the ontology. In order for all stakeholders to fully understand the acronyms used in your requirements, they must be declared in the ontology. Please avoid this acronym or contact your domain architect in order to insert this acronym into the ontology	List of acronyms in the ontology (included in the ontology terms)
Ambiguity	Ambiguous sentences make the requirement difficult to understand, and can provoke other stakeholders to understand something different than the idea initially planned by the author of the requirement. Ambiguous sentences are difficult to understand, as two stakeholders may understand different needs in the same requirement	Ambiguous words/expressions to be avoided: adequate, approximately, approximate, as a maximum, as a minimum, maximum, minimum, minimal, as appropriate, appropriate, reasonable, as possible, as required, bad, be able to, be capable of, best practices, best possible, better, capability of, capability to, close quickly, easy, easy to, easy to use, effective, efficient, fast, flexible, good, high performance, high speed, if practical, improved, maximize, minimize, medium-sized, optimize, optimal, optimum, nominal, normal, typical, typically, useable, suitable, not limited to, provide for, prompt, quick, quickly, rapid, reliable, routine, safe, slow, sufficient, sufficiently, timely, too, user friendly, user-friendly, versatile, worst, at least, enough, clearly, based on, some, any, several, many, many of, a lot of, a few, few, about, very nearly, manage, about, easily, close to, small, significant, vague, flexible, ancillary, relevant, routine, common, generic, customary, so far as is possible, as far as possible, as little as possible, as much as possible, if it should prove necessary, as necessary, necessary, all, any, both
Boilerplates matching	The structure (grammar) of your requirements must fulfill one of the agreed grammars (boilerplates). Doing so will increase the readability of the requirement and you will ensure that the automatic tool will perfectly understand the requirement.	Number of boilerplates matched
Chars between punctuation	If you write long sentences without punctuation marks, the requirement will be difficult to understand. You must introduce more punctuation marks in order to get more readable requirements.	Number of counted characters between punctuation marks.
Conditional	A requirement must be written in an assertive way. Whether a requirement is mandatory or not must be indicated as an attribute, and not by using conditional expressions.	Examples of conditional words/expressions: May be, May be not, Can, Can't, Cannot, Could, Couldn't, Could not, Should, Shouldn't, Should not, Ought to, Oughtn't, Ought not, Would
Connectors	Using too many connectors, in most of the cases, may mean that either your requirement is over specified or you are mixing two or more different needs into the same requirement.	Examples of connectors: and, and / or, and/or, or, as well as, but also, however, whether, meanwhile, whereas, on the other hand, otherwise, /
Dependencies	An object with too many dependences could be difficult to understand	Number of in and out links

Figure 6-2: Some of the current RQS metrics (RQA tool screenshot)

The figure 6-3 shows the quality of FLCS specification. In raw, we can see all the metrics available at the beginning of the CRYSTAL project.

For this FLCS specification which was written without using RQS tools, RQA consider to identify 45 high quality requirements, 539 medium quality requirements and 5 low level requirements. These results might not be relevant because we use the tool without any specific customization to Sagem context but this test allows verifying the installation of the tool, the process and analysis of the categorized “low level” requirements have allowed to identify sentences which were not requirements but clarifications and very ambiguous sentences.



Metric	High quality	High (%)	Medium quality	Medium (%)	Low quality	Low (%)	N/A	N/A (%)	Max.	Min.	Average	Std. Dev.	Best user
Acronyms	270	45,84 %	0	0,00 %	319	54,16 %	0	0,00 %	12,00	0,00	2,01	2,57	f073000
Ambiguous sente...	494	83,87 %	0	0,00 %	95	16,13 %	0	0,00 %	4,00	0,00	0,23	0,60	f073000, FT05357
Boilerplates match...	83	14,09 %	0	0,00 %	506	85,91 %	0	0,00 %	1,00	0,00	0,14	0,35	FX567069
Conditional mode	584	99,15 %	0	0,00 %	5	0,85 %	0	0,00 %	3,00	0,00	0,01	0,15	f073000, F552843,
Connectors	558	94,74 %	24	4,07 %	7	1,19 %	0	0,00 %	5,00	0,00	0,32	0,65	f073000, FT05357
Dependencies	589	100,00 %	0	0,00 %	0	0,00 %	0	0,00 %	0,00	0,00	0,00	0,00	f073000, FT05102
Design sentences	572	97,11 %	16	2,72 %	1	0,17 %	0	0,00 %	3,00	0,00	0,03	0,20	f073000, FT05357
Domain concepts	346	58,74 %	50	8,49 %	193	32,77 %	0	0,00 %	16,00	0,00	2,02	2,17	F554927
Domain verbs	139	23,60 %	63	10,70 %	387	65,70 %	0	0,00 %	9,00	0,00	0,54	0,88	F209800
Flow sentences	529	89,81 %	59	10,02 %	1	0,17 %	0	0,00 %	3,00	0,00	0,14	0,45	f073000, FT05357
Imperative mode	348	59,08 %	0	0,00 %	241	40,92 %	0	0,00 %	1,00	0,00	0,59	0,49	FT540099, F55492
Implicit sentences	586	99,49 %	2	0,34 %	1	0,17 %	0	0,00 %	4,00	0,00	0,06	0,30	f073000, FT05102
Incomplete sente...	587	99,66 %	0	0,00 %	2	0,34 %	0	0,00 %	1,00	0,00	0,00	0,06	f073000, F552190,
In-links	589	100,00 %	0	0,00 %	0	0,00 %	0	0,00 %	0,00	0,00	0,00	0,00	f073000, FT05102
Negative sentences	586	99,49 %	0	0,00 %	3	0,51 %	0	0,00 %	3,00	0,00	0,07	0,29	f073000, FT05102
Nesting levels	472	80,14 %	117	19,86 %	0	0,00 %	0	0,00 %	7,00	1,00	3,94	1,07	f073000, FT05357
Number of chars ...	490	83,19 %	94	15,96 %	5	0,85 %	0	0,00 %	209,00	5,00	45,94	30,78	f073000, FT05357
OLE objects	589	100,00 %	0	0,00 %	0	0,00 %	0	0,00 %	1,00	0,00	0,02	0,12	f073000, FT05102
Out-links	589	100,00 %	0	0,00 %	0	0,00 %	0	0,00 %	0,00	0,00	0,00	0,00	f073000, FT05102
Passive voice	526	89,30 %	0	0,00 %	63	10,70 %	0	0,00 %	2,00	0,00	0,11	0,34	f073000, FT05357
Rationale sentenc...	589	100,00 %	0	0,00 %	0	0,00 %	0	0,00 %	0,00	0,00	0,00	0,00	f073000, FT05102
Readability	552	93,72 %	11	1,87 %	26	4,41 %	0	0,00 %	91,00	0,00	14,07	7,95	FT054307, FT053
Speculative sente...	582	98,81 %	0	0,00 %	7	1,19 %	0	0,00 %	3,00	0,00	0,02	0,16	f073000, FT05357
Subjective senten...	589	100,00 %	0	0,00 %	0	0,00 %	0	0,00 %	0,00	0,00	0,00	0,00	f073000, FT05102
Text length (parag...	525	89,13 %	30	5,09 %	34	5,77 %	0	0,00 %	11,00	1,00	1,26	0,96	f073000, FT05357
Text length (words)	449	76,23 %	22	3,74 %	118	20,03 %	0	0,00 %	113,00	1,00	16,41	13,63	FT053577, F55492
Unclassified conc...	46	7,81 %	110	18,68 %	433	73,51 %	0	0,00 %	52,00	0,00	5,41	5,42	FX567069
Unclassified verbs	365	61,97 %	0	0,00 %	224	38,03 %	0	0,00 %	6,00	0,00	0,47	0,72	f073000, FT05357
Volatility (Changes...	589	100,00 %	0	0,00 %	0	0,00 %	0	0,00 %	0,00	0,00	0,00	0,00	f073000, FT05102
Volatility (Versioni...	573	97,28 %	14	2,38 %	2	0,34 %	0	0,00 %	23,00	0,00	0,67	1,82	f073000, F552190,

Figure 6-3: Requirements quality analysis summary for FLCS specification (RQA tool screenshot)

This tool is very easy to use and does not need so much development. We mainly will have to customize new metrics.

6.1.3 First steps with “RAT” tool

The RAT tool allows writing new requirements to be stored in DOORS database using on fly the same metrics as RQA.

The Figure 6-4 is a screenshot of RAT when writing environmental requirement.

After having selected a relevant pattern from knowledgeManager (in that case, it is a DO-160 Environment conditions), writer begins to specify the system (“Aircraft systems shall…”). Thanks to the pattern to be followed (in the window "Current boilerplates elements"), RAT suggests using terms "comply" or "meet" to go on writing the requirement.

The colour around the requirement indicate if it matches with the metrics or not ("red" indicate a low quality requirement, "yellow/orange" a medium quality requirement as on the figure, "green" a high quality requirement).

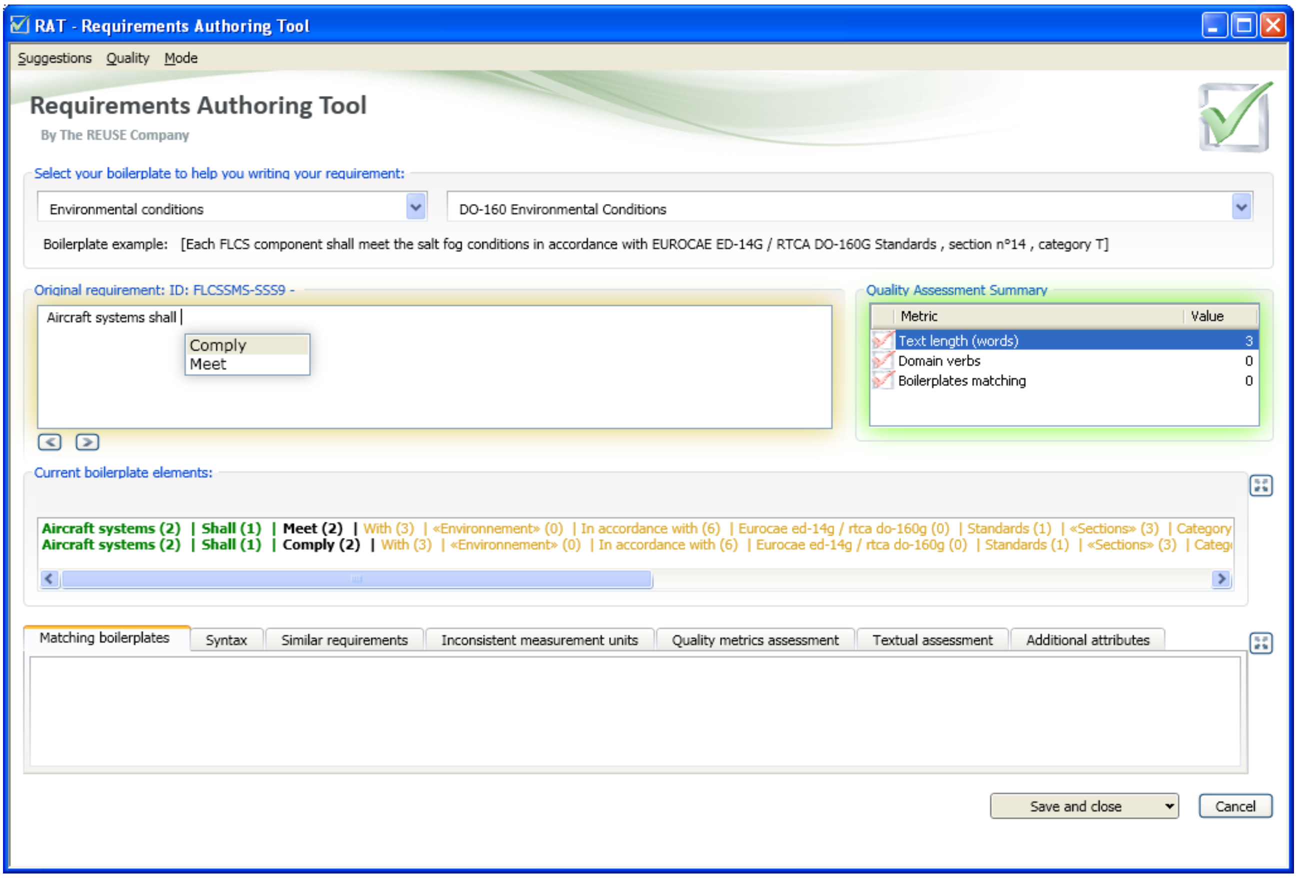


Figure 6-4: A requirement about environment compliance being written with RAT screenshot

This tool is very user friendly. Some tests have been performed to verify the capability to support collaborative work (2 users trying to write together in the same DOORS project). The tool seems to provide relevant warnings.

It should be improved within CRYSTAL project to provide another window with suggestion from previous project for similar requirements.

6.2 Preliminary evaluation of the tools capabilities with regards to CCC criteria

A preliminary estimation of the future capability of these tools to help satisfying the CESAR CCC criteria at the end of the CRYSTAL project has been performed. This estimation is quite promising so far and enabled to identify development activities for WP607 for defining advanced quality checks. A first evaluation of the tool chain with regard to CCC criteria will be provided in next version of this deliverable and will include the algorithms of OFFIS based on formalization requirements.

7 Conclusion

The work-package progress is compliant with initial development plan.

End users have defined the needs to be covered for the WP6.7 (Requirements Based Engineering).

Tool providers have given a first feedback of their capabilities to satisfy these needs.

This feedback is quite positive and allows forecasting significant improvements in requirements process.

The tool providers were able to exchange knowledge about their technologies and seem having reached a good comprehension of their respective tool capabilities and finding a way to integrate OFFIS algorithms in back office of TRC toolset. This integration has to be tested within second iteration phase.

After this first iteration phase and preliminary analysis, we are confident in the capability of the tool chain to progressively reach our objectives in the order of priority:

- 1- To define and share common vocabulary within the project/organization/supply chain
- 2- To enhance the quality of the specification at each step of development process by helping the engineering team to satisfy the completeness/consistency/correctness criteria of a specification
- 3- To help the reuse of requirements from previous project.
- 4- To use interoperability services (still be clarified and confirmed with WP6.1: IOS)
- 5- We also expect to connect SysML 29odeling tool to the process (if time and providers resources allow) in order to help:
  - populating specific project ontology from SysML models
  - assuring consistency between requirements and specification models

That is why the second iteration of the WP can begin without significant modifications with regard to technical annex.

8 Terms, Abbreviations and Definitions

Please add additional terms, abbreviations and definitions for your deliverable.

CRYSTAL	**Cr**itical S**YST**em Engineering **A**cce**L**eration
R	Report
P	Prototype
D	Demonstrator
O	Other
PU	Public
PP	Restricted to other program participants (including the JU).
RBE	Requirements Based Engineering
RE	Restricted to a group specified by the consortium (including the JU).
CO	Confidential, only for members of the consortium (including the JU).
WP	Work Package
SP	Subproject
CCC	Completeness, Consistency, Correctness
IOS	InterOperability Specification
FCS	Flight Control System
FLCS	Flap Control System
RBE	Requirements Based Engineering
FCS	Flight Control System
kM	knowledgeManager tool
RAT	Requirements Authoring Tool
RQA	Requirements Quality Analyzer tool
RQS	Requirement Quality Suite (including RAT, RQA, knowledgeManager and OFFIS tools)
TRC	The Reuse Company

Table 8-1: Terms, Abbreviations and Definitions

9 References

[Author, Year]	Authors; *Title*; Publication data (document reference)
[Malot - CESAR, 2012]	Malot M. – CESAR D_SP2_R3.3_M3_vol 4 : Completeness/Consistency/correctness
[Llorens, 2004]	Llorens J. *RSHP: an information representation model based on relationships or Studies in Fuzziness and Soft Computing*, Springer. Volume 159, 2004, pp. 221-253.

10 Annex I: Overview of methods reused from WP2.08

10.1 Method “Fault-tree Generation”

The general purpose of this engineering method is to generate fault-tree’s out of a given set of engineering data. This fault-tree will be used to assess the failure probability for a given system concept.

Engineering Method: UC208_GenerateFaultTrees_001					
Purpose: The safety designer would like to generate fault trees corresponding to a list of failure conditions.					
Comments:					
Pre-Condition		**Engineering Activities (made of steps)**		**Post-Condition**	
The safety data is stored in a safety in-house tool Dysfunctional models are available		1. In FT+, search list of Failure Conditions by applying service “Get Failure Condition List” 2. Request is forwarded to In-House Safety Tool 3. In-House Safety tool sends failure condition list 4. In FT+, search list of failure components by applying service “Get Failure Components List” 5. Request is forwarded to a safety modeling and analysis tool based on AltaRica Language 6. For each Failure Condition, the list of components which is linked to the Failure condition is sent to FT+ 7. In FT+, the fault-trees are generated		Fault-trees are generated	
Notes:		Notes:		Notes:	
Artefacts Required as inputs of the Activities		**Artefacts used internally within the Activities (optional)**		**Artefacts Provided as outputs of the Activities**	
Name	Dysfunctional Models (with appropriate detailed descriptions)			Name	Fault-Trees Model (with appropriate detailed descriptions)
Generic Type: (Tool or language independend type)	Dysfunctional Models			Generic Type: (Tool or language independend type)	Fault-Tree Model
Required Properties: (Information required in interactions between steps)	TBD			Provided Properties: (Information provided in interactions between	TBD

				steps)	
Description & Interoperability Additional Constraints:				Description & Interoperability Additional Constraints:	
Name	Safety Data (with appropriate detailed descriptions)			Name	TBD
Generic Type: (Tool or language independend type)	Safety Data			Generic Type: (Tool or language independend type)	TBD
Required Properties: (Information required in interactions between steps)	TBD			Provided Properties: (Information provided in interactions between steps)	TBD
Description & Interoperability Additional Constraints:				Description & Interoperability Additional Constraints:	

10.2 Method "Verify Design against Requirements"

The general purpose of this engineering method is to ensure that a given system concept does not violate the system requirements. This engineering method can occur at different phases of the system development process, such as preliminary concept evaluation or detailed concept definition.

It is assumed that several modelling and simulation tools are involved, each tool providing models for a different viewpoint.

To ensure that requirements are not violated, the models describing the system concepts must be identified, and for each type of requirement the corresponding values must be extracted from the right model (e.g. through simulation or calculation).

Engineering Method: UC208_VerifyDesignAgainstRequirements_001		
Purpose: The Requirements Engineer wants to check if a Design alternative meets a set of given requirements		
Comments:		
Pre-Condition	**Engineering Activities (made of steps)**	**Post-Condition**

Requirements constraining the de-icing system have been defined. Models describing the design alternative with information about components weight and pressure have been defined.	1. In a Design Exploration tool, the required weight and pressure values are requested 2. Request is forwarded to Doors 3. System Weight and Pressure Requirements are send back 4. In the Design Exploration Tool, service “request System Weight” is launched by Requirements Engineer 5. Request is forwarded to Papyrus 6. System Weight is calculated 7. System Weight result is sent back 8. In Design Exploration tool, service “request maximum pressure in System” is launched by Requirements Engineer 9. Request is sent to Papyrus 10. Papyrus sends model to Dymola for simulation 11. In Dymola, run pressure simulation model to determine max. pressure 12. In Dymola, send maximum pressure value 13. Results are sent to Design Exploration Tool 14. In Design Exploration tool, compare system weight requirement and pressure requirement with calculation and simulation results	As a result of the verification of Design against weight and pressure requirements, a status “Failed” or “Passed” for each requirement is defined.
Notes:	Notes:	Notes:
Artefacts Required as inputs of the Activities	**Artefacts used internally within the Activities (optional)**	**Artefacts Provided as outputs of the Activities**

Name	Requirements	Name		Name	Status "Passed" or "Failed" for each requirement.
Generic Type: (Tool or language independend type)	Requirements in natural language format	Type:		Generic Type: (Tool or language independend type)	Property of a requirement
Required Properties: (Information required in interactions between steps)	Values for metrics that are constraining the de-icing system concepts, such as maximum weight, or max. allowed pressure values for de-icing fluid reservoir.	Properties:	Values for metrics that are constraining the de-icing system concepts, such as maximum weight, or max. allowed pressure values for de-icing fluid reservoir.	Provided Properties: (Information provided in interactions between steps)	Status "Passed" or "Failed" for each requirement.
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	
Name	De-icing System Weight Model	Name		Name	
Generic Type: (Tool or language independend type)	Weight Model	Type:		Generic Type: (Tool or language independend type)	

Required Properties: (Information required in interactions between steps)	Overall System Weight	Properties:	Overall System Weight	Provided Properties: (Information provided in interactions between steps)	
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	
Name	De-icing System Physical Behavior Model	Name		Name	
Generic Type: (Tool or language independend type)	Physical Behavior Model (e.g. in Modelica or Matlab/Simulink)	Type:		Generic Type: (Tool or language independend type)	
Required Properties: (Information required in interactions between steps)	Resulting values for physical behavior simulation - this could be e.g. Resulting pressure value for a de-icing fluid reservoir.	Properties:	Resulting values for physical behavior simulation - this could be e.g. Resulting pressure value for a de-icing fluid reservoir.	Provided Properties: (Information provided in interactions between steps)	
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	

10.3 Method “Trade-off Analysis”

The general purpose of this engineering method is to compare different given system concepts with each other. This engineering method can occur at different phases of the system development process, such as preliminary concept evaluation or detailed concept definition.

It is assumed that several modelling and simulation tools are involved, each tool providing models for a different viewpoint for each of the alternative system concepts.

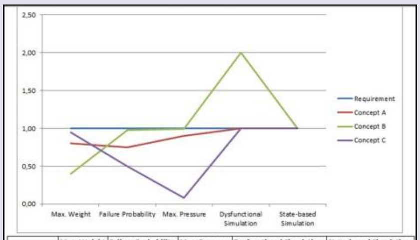
To compare the system concepts with each other, the relevant metrics for comparison have to be identified. Then, the models describing the system concepts must be identified, and for each type of metric the corresponding values must be extracted from the right model (e.g. through simulation or calculation).

Engineering Method: UC208_Trade-Off Analysis_001		
Purpose: The System Architect of the De-icing system wants to evaluate different alternative de-icing system concepts		
Comments: The concepts are described by many different models, each representing one or several viewpoints		
Pre-Condition	**Engineering Activities (made of steps)**	**Post-Condition**

The alternative concepts for the de-icing system are described by many different models, each representing one or several viewpoints

1. System Architect defines the metrics that are important for assessing a de-icing system concept (e.g.: weight, failure probability, max. pressure, max. response time, etc.)
2. System Architect launches request “Get Constraints”
3. Request is transferred to DOORS
4. Constraints for the de-icing system are sent back (or shown)
5. System Architect launches request “Assemble analysis results”
6. Request is transferred to tools that are storing data that is relevant to assess the de-icing model “Concept A”, “Concept B”, and “Concept C”
7. Tools are launching simulations and calculations
8. Tools are sending simulation and calculation results to Trade-off Tool

All simulation and calculation results are presented to the System Architect.
Example:



	Max. Weight	Failure Probability	Max. Pressure	Dysfunctional Simulation	State-based Simulation
Requirement	1000 kg	1.0 E-09	10 bar	Has to be OK	Has to be OK
Concept A	800 kg	7.5 E-10	9 bar	OK	OK
Concept B	400 kg	9.8 E-10	9.9 bar	NOK	OK
Concept C	950 kg	5 E-10	8 bar	OK	OK

Notes:	Notes:	Notes:

Artefacts Required as inputs of the Activities		**Artefacts used internally within the Activities (optional)**		**Artefacts Provided as outputs of the Activities**	
Name	Requirements	Name		Name	Trade-off Analysis Results
Generic Type: (Tool or language independend type)	Requirements in natural language format	Type:		Generic Type: (Tool or language independend type)	Comprehensive Representation of simulation and calculation results for each alternative de-icing system concept against pre-defined metrics
Required Properties: (Information required in interactions between steps)	Values for metrics that are constraining the de-icing system concepts, such as maximum weight, cost, failure propabilities, pressure values, required time for de-icing	Properties:	Values for metrics that are constraining the de-icing system concepts, such as maximum weight, cost, failure propabilities, pressure values, required time for de-icing	Provided Properties: (Information provided in interactions between steps)	Required Values (from Requirements) per Metric Provided Values (from different models) for each alternative system concept
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	
Name	De-icing System Concept Model	Name		Name	
Generic Type: (Tool or language independend type)	Logical Architecture Model with State-based behavior	Type:		Generic Type: (Tool or language independend type)	
Required Properties: (Information required in	State-based Simulation results + some simple static parameters such as	Properties:	State-based Simulation results + some simple static parameters such as	Provided Properties: (Information provided in	

interactions between steps)	purchase cost		purchase cost	interactions between steps)	
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	
Name	De-icing System Safety Model	Name		Name	
Generic Type: (Tool or language independend type)	Safety Model for Failure Rate Calculation and for dysfunctional Simulation	Type:		Generic Type: (Tool or language independend type)	
Required Properties: (Information required in interactions between steps)	Dysfunctional Simulation result + overall system failure rates (for loss and erroneous)	Properties:	Dysfunctional Simulation result + overall system failure rates (for loss and erroneous)	Provided Properties: (Information provided in interactions between steps)	
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	
Name	De-icing System Weight Model	Name		Name	
Generic Type: (Tool or language independend type)	Weight Model	Type:		Generic Type: (Tool or language independend type)	
Required Properties: (Information required in interactions between steps)	Overall System Weight	Properties:	Overall System Weight	Provided Properties: (Information provided in interactions between steps)	
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	
Name	De-icing System Physical Behavior Model	Name		Name	
Generic Type: (Tool or language independend type)	Physical Behavior Model (e.g. in Modelica or Matlab/Simulink)	Type:		Generic Type: (Tool or language independend type)	
Required Properties: (Information required in interactions between steps)	Resulting values for physical behavior simulation - this could be e.g. resulting ice accretion in mm, time for ice elimination in seconds, amount of consumed "goods" for ice elimination in kg, liter, or kWh, and many more.	Properties:	Resulting values for physical behavior simulation - this could be e.g. resulting ice accretion in mm, time for ice elimination in seconds, amount of consumed "goods" for ice elimination in kg, liter, or kWh, and many more.	Provided Properties: (Information provided in interactions between steps)	

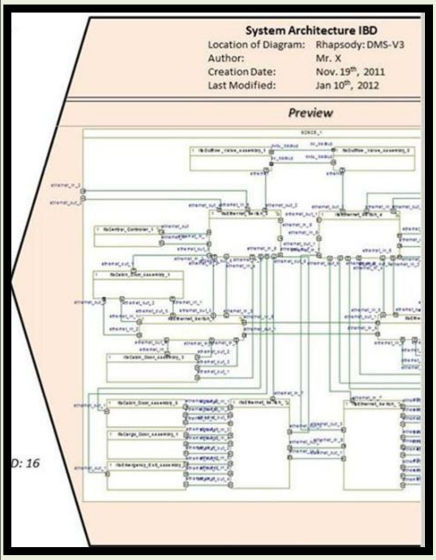
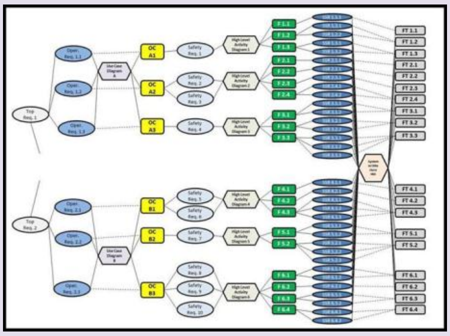
Description & Interoperability Additional Constraints:	Description:	Description & Interoperability Additional Constraints:

10.4 Method "Change Impact Analysis"

The general purpose of this engineering method is to assess the impact of a change in a top level requirement to a given system definition baseline. It is assumed that several tools are involved, each tool providing a different type of data that is relevant to describe the system definition baseline (e.g. derived requirements, models with different levels of granularity and for different viewpoints, simulation results, test results, implemented code, bill of materials, and other types of product documentation).

To assess the impact of a change in a top level requirement, all data elements have to be identified and presented to the engineer that are related to the top level requirement.

Engineering Method: UC208_ChangeImpactAnalysis_001		
Purpose: Requirements Engineer wants to assess the impact of a requirement change to the current technical solution of the de-icing system.		
Comments:		
Pre-Condition	**Engineering Activities (made of steps)**	**Post-Condition**

Engineers have defined a first technical solution for the de-icing system based on a given set of requirements. The technical solution is described in many different models managed by various tools and data-bases. A key requirement is changed.		1. Upon receival of a Change Request, Requirements Engineer changes the weight requirement and launches request to get list of related data objects 2. Request is forwarded to other tools 3. Tools are sending back list of related data 4. A "traceability" table or matrix is created to illustrate the related data 5. Requirements engineer requests a preview of a system architecture model that is related to the weight requirement 6. Request is forwarded to the respective modeling tool 7. Modeling tool is generating a preview of the architecture and sends it back. Example:  		Traceability matrix illustrated the data impacted by requirements change is created. Example: 	
Notes:		Notes:		Notes:	
Artefacts Required as inputs of the Activities		**Artefacts used internally within the Activities (optional)**		**Artefacts Provided as outputs of the Activities**	
Name	Requirements	Name		Name	Change Impact Results
Generic Type: (Tool or language independend type)	Requirements in natural language format	Type:		Generic Type: (Tool or language independend type)	Table, Document or Model
Required Properties: (Information required in interactions between steps)	Version, Baseline, Date of Creation, Approval Status, Author + Values for metrics that are constraining the de-icing system concepts and that are now being changed, such as maximum weight.	Properties:	Version, Baseline, Date of Creation, Approval Status, Author + Values for metrics that are constraining the de-icing system concepts and that are now being changed, such as maximum weight.	Provided Properties: (Information provided in interactions between steps)	Version, Baseline, Date of Creation, Approval Status, Author of all data and models that are impacted by the requirements change.
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	

Name	De-icing System Model	Name		Name	
Generic Type: (Tool or language independend type)	Logical Architecture Model with state-based behavior	Type:		Generic Type: (Tool or language independend type)	
Required Properties: (Information required in interactions between steps)	Version, Baseline, Date of Creation, Approval Status, Author, State-based Simulation results + some simple static parameters such as purchase cost	Properties:	Version, Baseline, Date of Creation, Approval Status, Author, State-based Simulation results + some simple static parameters such as purchase cost	Provided Properties: (Information provided in interactions between steps)	
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	
Name	De-icing System Safety Model	Name		Name	
Generic Type: (Tool or language independend type)	Safety Model for Failure Rate Calculation and for dysfunctional Simulation	Type:		Generic Type: (Tool or language independend type)	
Required Properties: (Information required in interactions between steps)	Version, Baseline, Date of Creation, Approval Status, Author, Dysfunctional Simulation result + overall system failure rates (for loss and erroneous)	Properties:	Version, Baseline, Date of Creation, Approval Status, Author, Dysfunctional Simulation result + overall system failure rates (for loss and erroneous)	Provided Properties: (Information provided in interactions between steps)	
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	
Name	De-icing Physical Behavior Model A based on Simulink	Name		Name	
Generic Type: (Tool or language independend type)	Physical Behavior Model (in Matlab/Simulink)	Type:	Model elements, especially diagrams	Generic Type: (Tool or language independend type)	
Required Properties: (Information required in interactions between steps)	Version, Baseline, Date of Creation, Approval Status, Author, Resulting values for physical behavior simulation - e.g. resulting ice accretion in mm.	Properties:	Version, Baseline, Date of Creation, Approval Status, Author, Resulting values for physical behavior simulation - e.g. resulting ice accretion in mm.	Provided Properties: (Information provided in interactions between steps)	
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	
Name	De-icing Physical Behavior Model B based on Dymola	Name		Name	
Generic Type: (Tool or language independend type)	Physical Behavior Model (e.g. in Modelica)	Type:		Generic Type: (Tool or language independend type)	

Required Properties: (Information required in interactions between steps)	Version, Baseline, Date of Creation, Approval Status, Author, Resulting values for physical behavior simulation - e.g. resulting ice accretion in mm.	Properties:	Version, Baseline, Date of Creation, Approval Status, Author, Resulting values for physical behavior simulation - e.g. resulting ice accretion in mm.	Provided Properties: (Information provided in interactions between steps)	
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	
Name	De-icing System Weight Model	Name		Name	
Generic Type: (Tool or language independend type)	Weight Model	Type:		Generic Type: (Tool or language independend type)	
Required Properties: (Information required in interactions between steps)	Version, Baseline, Date of Creation, Approval Status, Author, Overall System Weight	Properties:	Version, Baseline, Date of Creation, Approval Status, Author, Overall System Weight	Provided Properties: (Information provided in interactions between steps)	
Name	De-icing System Product Data	Name		Name	
Generic Type: (Tool or language independend type)	Files, Code, Documents, Models under configuration, etc.	Type:		Generic Type: (Tool or language independend type)	
Required Properties: (Information required in interactions between steps)	Version, Baseline, Date of Creation, Approval Status, Author.	Properties:	Version, Baseline, Date of Creation, Approval Status, Author.	Provided Properties: (Information provided in interactions between steps)	

10.5 Method “Maintain Consistency between multi-viewpoint models”

The general purpose of this engineering method is to ensure that a models describing a given system concept are consistent with each other.

Engineering Method: UC208_MaintainConsistencyBetweenMultiViewpointModels_001		
Purpose: Engineers want to ensure that their models are consistent (for those data that is used in many different tools) after a change occurs.		
Comments:		
Pre-Condition	**Engineering Activities (made of steps)**	**Post-Condition**
Engineers have defined many models to describe a technical solution for the de-icing system. Each model represents a different viewpoints of the de-icing system: - For example, a SysML tool could be used to describe the baseline architecture for the deicing system (logical or technical view) - For example, the AltaRica tool could used to define a model that describes the safety view of the system - For example, Matlab/Simulink could be used to define a model that describes the pressure view - For example, Papyrus could be used to define a weight model Some of the Models that describe the de-icing system contain data that is used by other models as well (e.g. a valve that regulates a de-icing fluid is used in the Safety Model and in the Pressure Model)	1. In SysML tool, the engineer managing the baseline model of de-icing system is changing Valve A (e.g. using a different Valve from another supplier). He launches the service “send data update” 2. The new data for the modified Valve A is forwarded to all other tools that are using Valve A in their models 3. Engineers working on other tools get the notification that the models are not consistent any more with the baseline, since Valve A has been changed 4. Engineers are accepting the update of the data in their models Alternative 1: Data would be automatically updated, and engineers would just get a respective notification Alternative 2: A Data Object “Valve A” does not physically exist in the models of the other engineers, they just have links to the original “Valve A” object. In that case, their models are also automatically updated as soon as the original data in baseline model changes.	All models are consistent with each other
Notes:	Notes:	Notes:

Artefacts Required as inputs of the Activities		**Artefacts used internally within the Activities (optional)**		**Artefacts Provided as outputs of the Activities**	
Name	De-icing System Baseline Architecture Model + related Data Objects	Name		Name	
Generic Type: (Tool or language independend type)	Logical or Technical Architecture Model and related data objects (e.g. components, interfaces)	Type:		Generic Type: (Tool or language independend type)	
Required Properties: (Information required in interactions between steps)	Data object type, Data object ID, Version, Baseline, Date of Creation, Approval Status, Author	Properties:	Data object type, Data object ID, Version, Baseline, Date of Creation, Approval Status, Author	Provided Properties: (Information provided in interactions between steps)	Data object type, Data object ID, Version, Baseline, Date of Creation, Approval Status, Author

Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	
Name	De-icing System Safety Model + related data objects	Name		Name	
Generic Type: (Tool or language independend type)	Safety Model and related data objects with safety properties	Type:		Generic Type: (Tool or language independend type)	
Required Properties: (Information required in interactions between steps)	Data object type, Data object ID, Failure Rate of Data object, Version, Baseline, Date of Creation, Approval Status, Author	Properties:	Data object type, Data object ID, Failure Rate of Data object, Version, Baseline, Date of Creation, Approval Status, Author	Provided Properties: (Information provided in interactions between steps)	Data object type, Data object ID, Failure Rate of Data object, Version, Baseline, Date of Creation, Approval Status, Author
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	
Name	De-icing Physical Behavior Model based on Simulink + related data objects	Name		Name	
Generic Type: (Tool or language independend type)	Physical Behavior Model and related data objects	Type:	Model elements, especially diagrams	Generic Type: (Tool or language independend type)	
Required Properties: (Information required in interactions between steps)	Data object type, Data object ID, physical behavior property of Data object (e.g. max. allowed pressure that can pass through a valve), Version, Baseline, Date of Creation, Approval Status, Author	Properties:	Data object type, Data object ID, physical behavior property of Data object (e.g. max. allowed pressure that can pass through a valve), Version, Baseline, Date of Creation, Approval Status, Author	Provided Properties: (Information provided in interactions between steps)	Data object type, Data object ID, physical behavior property of Data object (e.g. max. allowed pressure that can pass through a valve), Version, Baseline, Date of Creation, Approval Status, Author
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	
Name	De-icing System Weight Model + related data objects	Name		Name	
Generic Type: (Tool or language independend type)	Weight Model and related and data objects	Type:		Generic Type: (Tool or language independend type)	
Required Properties: (Information required in interactions between steps)	Data object type, Data object ID, weight of Data object, Version, Baseline, Date of Creation, Approval Status, Author	Properties:	Data object type, Data object ID, weight of Data object, Version, Baseline, Date of Creation, Approval Status, Author	Provided Properties: (Information provided in interactions between steps)	Data object type, Data object ID, weight of Data object, Version, Baseline, Date of Creation, Approval Status, Author

10.6 Method "Search Data"

The general purpose of this engineering method is to provide information about the data that describes a given system definition (e.g. models, requirements, product documentation), such as version, author, date of creation.

Engineering Method: UC208_SearchData_001					
Purpose: The System Architect of the De-icing system wants to visualize the history of a data (different versions of a data). He does not know in which tool the data is defined.					
Comments:					
Pre-Condition		**Engineering Activities (made of steps)**		**Post-Condition**	
Data describing the de-icing system has been defined (e.g. Requirements, different types of models, simulation results, test results, safety calculation results). The data is stored in many different data-bases. It is assumed that each set of data has an owner and a version.		1. In a dedicated search engine tool, launch service "get owner data version" 2. Request is forwarded to the tools that are managing data 3. Information about the data managed by the tools is sent back to the search engine. Only the tool which owns the searched data sends the data.		In the search engine tools, information about the data is displayed (e.g. version, owner, type of data)	
Notes:		Notes:		Notes:	
Artefacts Required as inputs of the Activities		**Artefacts used internally within the Activities (optional)**		**Artefacts Provided as outputs of the Activities**	
Name	De-icing System Baseline Architecture Model + related Data Objects	Name		Name	
Generic Type: (Tool or language independend type)	Logical or Technical Architecture Model and related data objects (e.g. components, interfaces)	Type:		Generic Type: (Tool or language independend type)	
Required Properties: (Information required in interactions between steps)	Data object type, Data object ID, Version, Baseline, Date of Creation, Approval Status, Author	Properties:	Data object type, Data object ID, Version, Baseline, Date of Creation, Approval Status, Author	Provided Properties: (Information provided in interactions between steps)	Data object type, Data object ID, Version, Baseline, Date of Creation, Approval Status, Author
Description & Interoperability Additional		Description:		Description & Interoperability Additional	

Constraints:				Constraints:	
Name	De-icing System Safety Model + related data objects	Name		Name	
Generic Type: (Tool or language independend type)	Safety Model and related data objects with safety properties	Type:		Generic Type: (Tool or language independend type)	
Required Properties: (Information required in interactions between steps)	Data object type, Data object ID, Failure Rate of Data object, Version, Baseline, Date of Creation, Approval Status, Author	Properties:	Data object type, Data object ID, Failure Rate of Data object, Version, Baseline, Date of Creation, Approval Status, Author	Provided Properties: (Information provided in interactions between steps)	Data object type, Data object ID, Version, Baseline, Date of Creation, Approval Status, Author
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	
Name	De-icing Physical Behavior Model based on Simulink + related data objects	Name		Name	
Generic Type: (Tool or language independend type)	Physical Behavior Model and related data objects	Type:	Model elements, especially diagrams	Generic Type: (Tool or language independend type)	
Required Properties: (Information required in interactions between steps)	Data object type, Data object ID, physical behavior property of Data object (e.g. max. allowed pressure that can pass through a valve), Version, Baseline, Date of Creation, Approval Status, Author	Properties:	Data object type, Data object ID, physical behavior property of Data object (e.g. max. allowed pressure that can pass through a valve), Version, Baseline, Date of Creation, Approval Status, Author	Provided Properties: (Information provided in interactions between steps)	Data object type, Data object ID, Version, Baseline, Date of Creation, Approval Status, Author
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	
Name	De-icing System Weight Model + related data objects	Name		Name	
Generic Type: (Tool or language independend type)	Weight Model and related and data objects	Type:		Generic Type: (Tool or language independend type)	
Required Properties: (Information required in interactions between steps)	Data object type, Data object ID, weight of Data object, Version, Baseline, Date of Creation, Approval Status, Author	Properties:	Data object type, Data object ID, weight of Data object, Version, Baseline, Date of Creation, Approval Status, Author	Provided Properties: (Information provided in interactions between steps)	Data object type, Data object ID, Version, Baseline, Date of Creation, Approval Status, Author

10.7 Method “Provide Specification”

The general purpose of this engineering method is to identify and assemble all data needed for a system specification. It is assumed that the relevant data is managed by many different tools.

Engineering Method: UC208_ProvideSpecificationDocument_001					
Purpose: Responsible Engineer at OEM level wants to send specification for a sub-system to a supplier					
Comments:					
Pre-Condition		**Engineering Activities (made of steps)**		**Post-Condition**	
Data for the sub-system specification is spread among different tools		1. In all tools where specification relevant data is stored, engineers identify and tag the relevant data 2. Engineer launches service “get data for specification” 3. Request is forwarded to all relevant tools 4. From tools, specification relevant data is send back 5. In the main tool of the engineer, all data is assembled to one specification and send to supplier 6. Supplier receives specification ALTERNATIVE 5b. Supplier requests specification 6.b Specification is send upon request to supplier		Sub-System specification has been assembled and sent to supplier	
Notes:		Notes:		Notes:	
Artefacts Required as inputs of the Activities		**Artefacts used internally within the Activities (optional)**		**Artefacts Provided as outputs of the Activities**	
Name	Requirements	Name		Name	Subsystem Specification
Generic Type: (Tool or language independend type)	Requirements in natural language format	Type:	Set of Requirements	Generic Type: (Tool or language independend type)	Document or Model
Required Properties: (Information required in interactions between steps)	Version, Baseline, Date of Creation, Approval Status, Author	Properties:		Provided Properties: (Information provided in interactions between steps)	Version, Baseline, Date of Creation, Approval Status, Author
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	
Name	De-icing System Model	Name		Name	
Generic Type: (Tool or language independend type)	Logical Architecture Model	Type:	Model elements, especially diagrams	Generic Type: (Tool or language independend type)	

Required Properties: (Information required in interactions between steps)	Version, Baseline, Date of Creation, Approval Status, Author	Properties:		Provided Properties: (Information provided in interactions between steps)	
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	
Name	De-icing System Safety Model	Name		Name	
Generic Type: (Tool or language independend type)	Safety Model	Type:	Model elements, especially diagrams	Generic Type: (Tool or language independend type)	
Required Properties: (Information required in interactions between steps)	Version, Baseline, Date of Creation, Approval Status, Author	Properties:		Provided Properties: (Information provided in interactions between steps)	
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	

10.8 Method “Put all data under configuration control”

The general purpose of this engineering method is to put under Configuration Control the “functional view” (As Required, As Conceived) system solution once the concept is defined.
This engineering method aims to manage all the items related with the aeronautical product in the PLM tools, including the main output of the engineering conceptual and design phase, more specifically Requirements, Functions, System Elements (ASD S1000D compliance), Logical and Physical Architecture.
The System Architect / Configuration Manager will be able to import specific data and information (activities mapped to the functions, blocks mapped with the system) from the Rhapsody Model in order to be managed under configuration control and to support traceability all along the Product Life Cycle.
this activity shall be carried out in an integrated way with the following already available PLM view (as Designed, As Planned).

Engineering Method: UC208_Put all data under configuration control		
Purpose: CM wants to put under Configuration Control the “functional view” in order to manage and reuse these artifacts for similar product/capability classes.		
Comments:		
Pre-Condition	**Engineering Activities (made of steps)**	**Post-Condition**

Functions are managed as Activities of Activity Diagrams in MBSE Tool (SysML modeling). Systems/Sub-Systems/Logical Equipment are managed by Blocks in MBSE tool (SysML modeling). Systems/Subsystems/Logical Equipment/ Functions are managed as Configuration Item in PLM Tool. A Functional Specification defined in a SysML Model has been frozen as Baseline at the end of Functional Analysis (Black box activity diagram) applicable to a specific configuration A Functional Specification defined in a SysML Model has been frozen as Baseline at the end of Design Synthesis (White box activity diagrams) applicable to a specific configuration		AT SYSTEM LEVEL 1. In PLM, select the SYSTEM under analysis 2. In PLM, launch service “Get List of System Functionalities” 3. Request is forwarded to MBSE Tool (SysML modeling) 4. List of all functions is assembled and send back to PLM tool 5. In PLM, receive functions 6. In PLM, the developer associates information related to applicability to the imported Functions 7. In PLM, correlate System View CI to As-Designed View CI AT SUBSYSTEM LEVEL 1. In PLM, select the SYSTEM under analysis 2. In PLM, launch service “Get List of All Syb-system Functionalities” 3. Request is forwarded to MBSE (SysML modeling) 4. For each SUBSYSTEM the List of allocated functions is assembled and send back to PLM tool 5. In PLM, receive SUBSYSTEM functions 6. In PLM, the developer associates information related to applicability to the imported Functions		Systems View Management in PLM tool with Applicability management of Functionalities defined in MBSE tool (SysML modelling). Management of Commonalities and Comparison of different Functional Configuration in PLM. Management of traceability from System View CI to As-Designed View CI (e.g. Function to Part Number) in the PLM tool.	
Notes:		Notes:		Notes:	
Artefacts Required as inputs of the Activities		**Artefacts used internally within the Activities (optional)**		**Artefacts Provided as outputs of the Activities**	
Name	De-Icing System Model	Name		Name	
Generic Type: (Tool or language independend type)	Functional behavior model and Logical Architecture (Rhapsody - SysML)	Type:		Generic Type: (Tool or language independend type)	
Required Properties: (Information required in interactions between steps)	Version, Baseline, Activity Diagram (System / Subsystem Primitive Operations and Event), Internal Block Diagram, Block Definition Diagram	Properties:	Version, Baseline, Activity Diagram (System / Subsystem Primitive Operations and Event), Internal Block Diagram, Block Defition Diagram	Provided Properties: (Information provided in interactions between steps)	System View (System Functionalities list) System View links to other views (i.e “as designed view”)
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	
Name	De-Icing System Product Data	Name		Name	
Generic Type: (Tool or language	Files, Codes, Document, Models under configuration,	Type:		Generic Type: (Tool or	

independend type)	Functional Specification Baseline			language independend type)	
Required Properties: (Information required in interactions between steps)	Version, Baseline, Date of Creation, Approval Status, Author	Properties:		Provided Properties: (Information provided in interactions between steps)	
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	