PROPRIETARY RIGHTS STATEMENT

THIS DOCUMENT CONTAINS INFORMATION, WHICH IS PROPRIETARY TO THE CRYSTAL CONSORTIUM. NEITHER THIS DOCUMENT NOR THE INFORMATION CONTAINED HEREIN SHALL BE USED, DUPLICATED OR COMMUNICATED BY ANY MEANS TO ANY THIRD PARTY, IN WHOLE OR IN PARTS, EXCEPT WITH THE PRIOR WRITTEN CONSENT OF THE CESAR CONSORTIUM THIS RESTRICTION LEGEND SHALL NOT BE ALTERED OR OBLITERATED ON OR FROM THIS DOCUMENT. THE RESEARCH LEADING TO THESE RESULTS HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S SEVENTH FRAMEWORK PROGRAM (FP7/2007-2013) FOR CRYSTAL – CRITICAL SYSTEM ENGINEERING ACCELERATION JOINT UNDERTAKING UNDER GRANT AGREEMENT N° 332830 AND FROM SPECIFIC NATIONAL PROGRAMS AND / OR FUNDING AUTHORITIES.



CRitical SYSTem Engineering AcceLeration

Architecture of the Tool Chain for the Multi-Mode Navigation System

D206.021



DOCUMENT INFORMATION

Project	CRYSTAL
Grant Agreement No.	ARTEMIS-2012-1-332830
Deliverable Title	Architecture of the Tool Chain for the Multi-Mode Navigation System
Deliverable No.	D206.021
Dissemination Level	RE
Nature	R
Document Version	V3.0
Date	2014-04-30
Contact	Tomáš Kratochvíla
Organization	HON
Phone	+420 532 115 530
E-Mail	Tomas.Kratochvila@honeywell.com



AUTHORS TABLE

Name	Company	E-Mail
Nikola Beneš	Honeywell	Nikola.Benes@honeywell.com
Jan Beran	Honeywell	Jan.Beran@honeywell.com
Vít Koksa	Honeywell	Vit.Koksa@honeywell.com
Tomáš Kratochvíla	Honeywell	Tomas.Kratochvila@honeywell.com

REVIEW TABLE

Version	Date	Reviewer
1.0	11.4.2014	Jan Beran (internal review)
2.0	29.4.2014	Anne Monceaux (external reviewer)
3.0	30.4.2014	Jiří Barnat (external reviewer)



CONTENT

										NAVIGATION
										NAVIGATION
1										6
										6 6
2	ARC	HITECTU	RE C	OF THE	Ε ΟΝΤΟ	LOGY E	NGINE	ERIN	G TOOL CHAIN	7
2	2.2 E 2.2.1 2.2.2 2.2.3 2.2.4	ENTERPRIS SQL so Automa Tracea Examp	E ARC cripts. ation bility le	SHITECT Scripts view	C SCRIPT	S				
3									AIN	
4									•••••	
Z		Automa	ation	Plan, l	Request	and Res	sult		Selection	24
5	CON		I	•••••			•••••		•••••	
6	TER	MS, ABBI	REVI	ATION	IS AND	DEFINIT	IONS.		•••••	
7	REF	ERENCES	5				•••••		••••••	



Content of Figures

Figure 1 – Ontology Engineering Tool Chain	7
Figure 2 – Activity diagram of requirement formalization.	8
Figure 3 – The GUI of Lexiana tool	9
Figure 4 – Creation of the thesaurus in the Enterprise Architect	10
Figure 5 – output of copyDependenciesWithSentences script	12
Figure 6 – Work with automation scripts in Enterprise Architect	13
Figure 7 – Traceability view on the left and the corresponding elements in the diagram	15
Figure 8 – A diagram generated by Lexiana as shown in the Enterprise Architect	16
Figure 9 – Resulting UML ontology after manual modifications, part 1	17
Figure 10 – Resulting UML ontology after manual modifications, part 2	18
Figure 11 – Thesaurus semantics of the relationships between terms	19
Figure 12 – Development and verification tool chains together.	21
Figure 13 – Verification tool chain architecture with the verification methods	23
Figure 14 – Integration of ForReq tool and automation servers based on OSLC	24
Figure 15 – Relationships among of the OSLC automation specification resources	25

Content of Tables

Table 4-1: Terms	s, Abbreviations and Definitions	29
------------------	----------------------------------	----



1 Introduction

1.1 Role of deliverable

The purpose of this document is to define the architecture of the tool chain that will be used to develop multi-mode navigation system. The engineering methods are described in Crystal Deliverable D206.010.

1.2 Structure of this document

We have divided the tool chain architecture into 3 parts. Each part is described in its own chapter:

- 2 Architecture of the Ontology Engineering Tool Chain creates and manages the domain ontology. This tool chain is used by domain ontology experts only and is not supposed to be done for every project, since many projects will share the same domain ontology.
- **3** Architecture of the Development Tool Chain model-based development process tool chain is supposed to be used for every project. It uses the domain ontology for a given domain for initial requirement authoring. We use Honeywell model-based development called 3 View System Engineering process.
- **4** Architecture of Verification Tool Chain verification and validation of the artefacts developed by the development tool chain. We are focused on automated formal verification.



2 Architecture of the Ontology Engineering Tool Chain

This tool chain is used by domain ontology experts only. The idea is that the domain ontology is created once for each domain and only limited work is needed to manage the domain ontology to be up to date with the current state of the domain. This process is not supposed to be performed for each project, since many projects will share the same domain ontology.

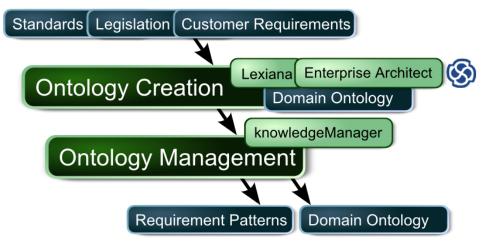


Figure 1 – Ontology Engineering Tool Chain

The tool chain will create ontologies for different domain granularities – from general ontology and aerospace ontology to specific domain ontologies (for example AHRS domain only). Since there are multiple approaches how to semi-automatically merge domain ontologies [Kotis, 2006], hence it will be possible to create exactly the ontology for any specific domain. For example, when the system under development integrates two subsystems into one, the ontologies for the two domains of the subsystems will be merged. Every domain ontology should have completely defined its domain using metadata showing which sources were used to create it. The general ontology might be used to show differences between more specific ontologies only.

Honeywell has a special tool suite consisting of Lexiana and Enterprise Architect scripts to semi-automatically derive requirements model from domain descriptions: standards, legislation, customer requirements, etc. Lexiana generates a list of suggested concepts and relations between the concepts from the domain descriptions. Then the domain ontology expert imports the requirement model into Enterprise Architect and makes the requirement model compact and consistent using several specialized Honeywell scripts. These compact requirement models will be then translated to the thesaurus.



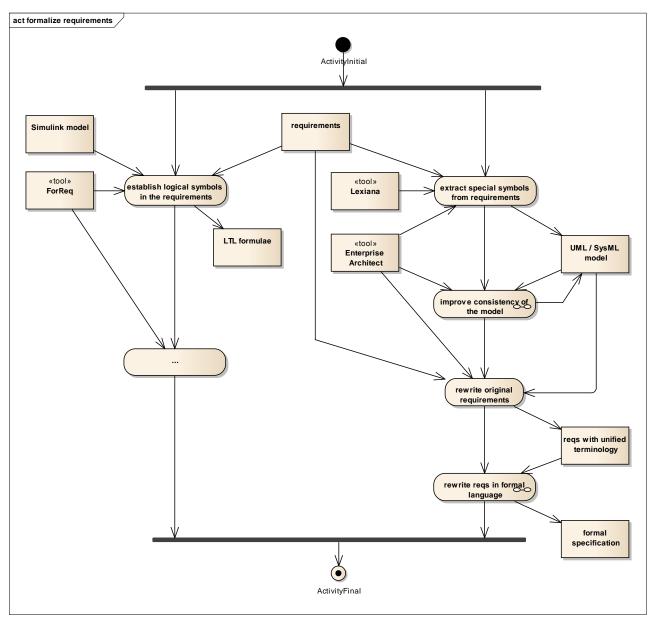


Figure 2 – Activity diagram of requirement formalization.

2.1 Lexiana

The extraction of interesting concepts from the text is supported by the tool Lexiana. This tool calls the Stanford Parser, which for English sentences generates their grammatical parsing trees. The grammatical parsing trees are processed by Lexiana. The output of Lexiana is an XMI file with the UML model of the extracted information, i.e.

 classes (their colours indicate the abundance of the term in the text – from green, which corresponds to 1 occurrence, to magenta, which corresponds to the most occurrences)

Version	Nature	Date	Page
V03.00	R	2014-04-30	8 of 30



- packages of classes (package sizes suitable for further manual processing, packaging based simply on the initial letters of the class names)
- tentative relations between the classes,
- original sentences,
- realization relations from classes to sentences (this kind of traces is useful throughout the evolution of the model, but can also help to easily make the original text more consistent by using a unified terminology, see the SQL script copyDependenciesWithSentences below),
- diagrams with the most frequent concepts,
- diagrams with concepts whose names contain common substrings.

😲 Lexiana v.1.3	
C:/Temp/lexiana/input_DO334str09_18.txt Choose Choo	Exit
Parse text grammatically Extract UML elements Choose	
. \outNlp.txt	
<pre>the, test, conditions, specified, within, this, document, .] Parsing [sent. 190 len. 12]: [Static, condition, is, not, necessarily, equivale t, to, straight, and, level, flight, .] Parsing [sent. 191 len. 20]: [Strapdown, -, Refers, to, a, system, that, uses, irect, mounting, of, sensors, -LRB-, without, gimbals, -RRB-, to, a, vehicle, Parsing [sent. 192 len. 20]: [Transport, Rate, -, An, apparent, angular, rate, f, the, aircraft, due, to, motion, over, the, surface, of, the, earth, .] Parsing [sent. 193 len. 11]: [True, Heading, -, Aircraft, heading, defined, rel tive, to, True, North, .] Parsing [sent. 194 len. 16]: [True, North, -, The, direction, along, the, earth s, surface, toward, the, geographic, North, Pole, .] Parsing [sent. 195 len. 33]: [Truth, Reference, -, An, AHRS, or, navigation, sy tem, of, which, heading, and, attitude, outputs, are, assessed, to, be, more, a curate, than, the, unit, being, tested, by, a, least, a, factor, of, two, .] Parsing [sent. 197 len. 29]: [Vs, -, Stalling, speed, or, minimum, steady, flig t, speed, at, which, the, airplane, is, controllable, Vso, -, Stalling, speed, r, minimum, steady, flight, speed, in, the, landing, configuration, .] Parsed file: C:/Temp/lexiana/input_D0334str09_18.txt [197 sentences]. Parsing done. Extraction finished. (466 classes generated)</pre>	d o la l, ys ac , yh





The classes are generated from certain noun phrases. For each sentence, the anonymous undirected relationships are generated between each two classes contained within the sentence. Most of these tentative relationships are usually removed later during the manual processing of the model.

2.2 Enterprise Architect Scripts

The Enterprise Architect (EA) is a robust tool for UML or SysML modelling. The requirements model generated by Lexiana is imported into EA. The generated UML model (the imported elements and diagrams) is a good starting point and spares time of the analyst, but this model still usually needs substantial improvements. The creation of a mature ontology from the original model is facilitated by several utilities, which were incorporated as either SQL scripts or automation scripts within the Enterprise Architect. Some of these scripts provide useful information on the model, while other scripts are called to update the model. The scripts can speed up some activities by orders of magnitude, remove tedious work and help to avoid omissions.

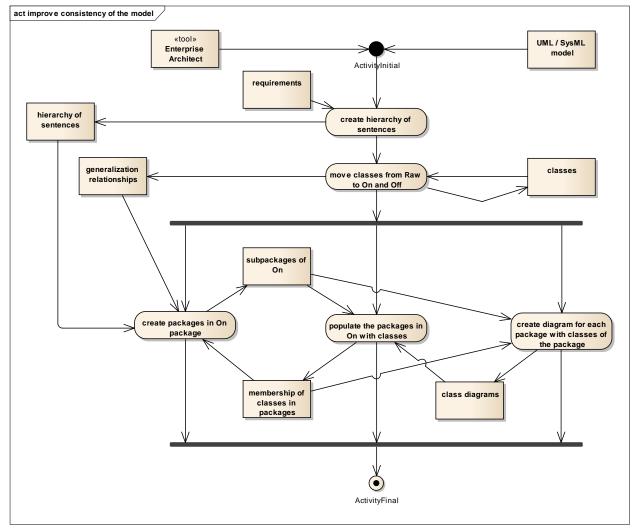


Figure 4 – Creation of the thesaurus in the Enterprise Architect

Version	Nature	Date	Page
V03.00	R	2014-04-30	10 of 30



In the figure above the following activities are depicted:

- **Create hierarchy of sentences** the requirements are packaged according to the original document structure.
- Move classes from Raw to On and Off originally, all classes are under the package Raw; move them to either On (important classes) or Off (thrown out classes). The most useful automation scripts (see below) are Lxn-copyLinks for merging or splitting classes, and Lxn-listReqsOfClass for listing the related requirements. The generated "stem" diagrams are indispensable for this activity, as they display together the classes with common substrings.
- Create packages in On package usually there is too much classes in the On package. Subpackages can be e.g. generated by the Lxn-mirrorReqPackageToObjPackage automation script (see below), or created around classes which have numerous subclasses.
- **Populate the packages in On with classes** when the structure of packaging was mirrored from the packaging of the requirements, the Lxn-addInhabitants automation script can help to move the relevant classes to the appropriate package.
- create diagram for each package with classes of the package place the classes of the package into the diagram of the package. Use the scripts Lxn-addTentativeNeighbours, Lxn-addNeighboursOfClass, Lxn-addSuperclassesOfClass to populate the diagram.

2.2.1 SQL scripts

Most of the SQL scripts are generally usable. The SQL scripts query the database with the model. No modifications of the database content are allowed via this mechanism.

Example of what the **copyDependenciesWithSentences** script produces is in shown in the following figure.

JYXY
CRYSTAL
775

e Edit View Project Diagram				
🗳 • 🗄 🕺 🕼 📽 🖉 • 🍋	. 🗋 🐡 🗖 🔇	Basic UML® 2 Technol 💌	🎯 🚽 🛄 🚰 Bas	sic - Normal 🔄 🖕 🗄 ManchesterOWL 💌 <none> 💌 🕾 Import 👻 🎧 🚱 🚱 🍪 🖕</none>
		• 미 🕼 역 속 비 🖄 🖏	\$3 a .	
Model Search				* •
B 🗙 🛱 🗟 🖓 • 🖉 🎯				
Search Term: 90	Search: copys	StereotypSVetama 👻	Bun Options	s 🔻 Builder 🔺
° 9 8 @ ¢ × °r *r 18 8				
SELECT ono.Name AS Final, c.Btm FROM t connector AS c, t object				
WHERE EXISTS		ie no ono, conject no regan,		
	0.Object Type	AS CLASSTYPE o.Object ID as	ID o.Object Type	as Type,o.Name as Name,o.Note as Documentation,o.Package JD as ParentID,p.name as ParentName,
				", p7Name b) rame, bit voice as bocomentation (or ackage) b) rame b) rame b) rame voice as rame, "
				", p3.Name + ',') + "If(isNull(o2.Name), ", p2.Name + ',') + p.Name + ',' + o.Name) as OualifiedName
FROM ((((((((t object as o	i y · m(isi vui(pa	and an example of the second s	(is) vangesti vanne),	, pontance - 17 - antisticingesticance, , pesticance - 17 - pricance - 17 - ontraine) do Qualmeuraline
left join t package p on o.Packag	e ID - p Package	(D)		
left join t_package p2 on p.Paren				
	icio – pziedokag	e_10)		
Drag a column header here to group by that	t column.			····
	t column. Btm_Mid_Label	Superceded	SentenceName	Sertence
Drag a column header here to group by tha		Superceded AhrsInertialSystem	SentenceName aSentence 00142	Sertence Estemal Adrig - Any source of information provided from a system external to the AHRS inertial system or function.
Drag a column header here to group by tha	Btm_Mid_Label			
Vrag a column header here to group by the nal titude And Heading Reference System titude And Heading Reference System	Btm_Mid_Label	AhrsInertialSystem	aSentence 00142	External Aiding - Any source of information provided from a system external to the AHRS inertial system or function.
Drag a column header here to group by the inal Ittude And Heading Reference System Ittude And Heading Reference System Ittude And Heading Reference System	Btm_Mid_Label «copy» «copy»	AhrsInertialSystem AhrsSystem	aSentence 00142 aSentence 00068	Edemail Ading - Any source of information provided from a system external to the AHRS inertial system or function. An AHRS system may have multiple modes of operation.
Drag a column header here to group by the inal Ittude And Heading Reference System Ittude And Heading Reference System Ittude And Heading Reference System Ittude And Heading Reference System	Btm_Mid_Label «copy» «copy» «copy»	AhrsInertialSystem AhrsSystem AhrsSystem	aSentence 00142 aSentence 00068 aSentence 00028	Edenal Adrog - Any source of information provided from a system edemal to the AHRS inetial system or function. An AHRS system may have multiple modes of operation. A admonin Figure 11: the AHRS system may consist of: 12 Altitude and Heading Reference Unit (AHRU).
Ang a column header here to group by the nal tude And Heading Reference System ttude And Heading Reference System ttude And Heading Reference System ttude And Heading Reference System	Btm_Mid_Label «copy» «copy» «copy» «copy»	AhrsInertialSystem AhrsSystem AhrsSystem AhrsTum	aSentence 00142 aSentence 00068 aSentence 00028 aSentence 00055	Edemai Adrog - Any source of infomation provided from a system external to the AHRS inertial system or function. An AHRS system may have multiple modes of operation. As shown in Figure 1-1, the AHRS system may consist of: 1 2.1 Atstude and Heading Reference Unit (AHRU). The AHRS Lim and dip performance requirements are provided in Section 2.25.
Ang a column header here to group by this nail titude And Heading Reference System titude And Heading Reference System titude And Heading Reference System titude And Heading Reference System titude And Heading Reference System	Btm_Mid_Label «copy» «copy» «copy» «copy» «copy» «copy»	AhrsInertialSystem AhrsSystem AhrsSystem AhrsTum DistinguishingAhr	aSentence 00142 aSentence 00068 aSentence 00028 aSentence 00055 aSentence 00062	Edemal Ading - Any source of information provided from a system external to the AHRS inetial system or function. An AHRS system may have multiple modes of operation. As a shown in Figure 1-1, the AHRS system may consult of: 12.1 Attrude and Heading Reference Unit (AHRU). The AHRS turn and slip performance requirements are provided in Section 22.5. Different accuracy calepoides have been specified to help in distriguishing AHRS of different capabilities.
Integration of the second seco	Btm_Mid_Label «Copy» «Copy» «Copy» «Copy» «Copy» «Copy» «Copy»	AhrshertialSystem AhrsSystem AhrsSystem AhrsTum DistinguishingAhr OrMoreAhr	aSentence 00142 aSentence 00068 aSentence 00028 aSentence 00055 aSentence 00062 aSentence 00036	Extensi Adrog - Any source of information provided from a system external to the AHRS inetial system or function. An AHRS system may have multiple modes of operation. A shown in Figure 11-11 we AHRS system may consist of 1.2.1 Kittude and Heading Reference Unit (AHRU). The AHRS turn and silp performance requirements are provided in Section 2.2.5. Different accuracy categories have been specified to help in distinguishing AHRS of different capabilities. One or more AHRS may be used on an airceful to apply categories of the aircraft.
Carg a column header here to group by the mail thuck-hrd Heading/Reference System thuck-hrd Heading/Reference System thuck-hrd Heading/Reference System thuck-hrd Heading/Reference System thuck-hrd Heading/Reference System thuck-hrd Heading/Reference System thuck-hrd Heading/Reference System	Btm_Md_Label «copy» «copy» «copy» «copy» «copy» «copy» «copy»	AhrshertialSystem AhrsSystem AhrsSystem AhrsTum DistriguishingAhr OrMoreAhr Ahru	a Sentence 00142 a Sentence 00068 a Sentence 00028 a Sentence 00055 a Sentence 00052 a Sentence 00036 a Sentence 00120	Edemai Adrog - Any source of information provided from a system external to the AHRS inertial system or function. An AHRS system may have multiple modes of operation. As shown in Figure 1-1, the AHRS system may consist of: 1.2.1 Atitude and Heading Reference Unit (AHRU). The AHRS Lima and dip performance requirements are provided in Section 2.2.5. Different accuracy categories have been specified to help in distinguishing AHRS of different capabilities. One or more AHRS may be used on an aircraft to supply atitude and heading information to the pilots of the aircraft to aid them with the control and guidance of the aircraft. AHRU -Attitude and Heading Reference Unit.
hang a column header here to group by the mail thuck-hold Heading Performs System thuck-hold Heading Performs Dist thuck-hold Heading Performs Dist thuck-hold Heading Performs Dist thuck-hold Heading Performs Dist	Btm_Md_Label «COpy» «COpy» «COpy» «COpy» «COpy» «COpy» «COpy» «COpy» «COpy»	AhrsInertialSystem AhrsSystem AhrsSystem AhrsTum DistinguishingAhr OrtforeAhr Ahru Ahru	a Sentence 00142 a Sentence 00068 a Sentence 00028 a Sentence 00052 a Sentence 00052 a Sentence 00036 a Sentence 00120 a Sentence 00020	External Adrog - Any source of information provided from a system external to the AHRS inertial system or function. An AHRS system may have multiple modes of operation. A shown in Figure 11: the AHRS system may constat of: 12.1 Kittude and Heading Reference Unit (AHRU). The AHRSE turn and sho performance requirements are provided in Sector 22.5. Different accuracy calegories have been specified to help in deingualiting AHRS of different capabilities. One or more AHRS may be used on an aroaft to supply situade and heading information to the pilots of the aircraft to ad them with the control and guidance of the aircraft. AHRU - Astructe and Heading Reference Unit.
Any a column header here to group by the man and to the schroll Reference System touck-And Heading Reference Unit touck-And Heading Reference Unit	Btm_Mid_Label «copy»	AhrsinetialSystem AhrsSystem AhrsSystem AhrsTum DistinguishingAhr OrMoreAhr Ahru Ahru Ahru	a Sentence 00142 a Sentence 00068 a Sentence 00055 a Sentence 00055 a Sentence 00036 a Sentence 00120 a Sentence 00020 a Sentence 00023	Edemai Adrog - Any source of information provided from a system external to the AHRS inertial system or function. An AHRS system may have multiple modes of operation. As abrown in Figure 1-1 the AHRS system may consist of 1.2.1 Attrude and Heading Reference Unit (AHRU). The AHRS turn and silp performance requirements are provided in Section 2.2.5. Different accuracy categories have been specified to help in distinguishing AHRS of different capabilities. One or more AHRS may be used on an architot ta key by attrude and heading information to the pilots of the aircraft to add them with the control and guidance of the aircraft. AHRU - Attrude and Heading Reference Unit. This document consides an acquipment configuration consisting of an Attrude and Heading Reference Unit (AHRU) with complementary peripheral equipment specified by the AHRS manufacturer. For example, the "equipment" may include an AHRU, an associated mounting yru, and a magretic exercising unit.
Ang a column header here to group by the mail titude And Heading Reference System titude And Heading Reference Lint titude And Heading Reference Lint titude And Heading Reference Lint titude And Heading Reference Lint	Btm_Md_Label eccopys	AhrsinettialSystem AhrsSystem AhrsSystem AhrsSystem AhrsUm DistingJakhngAhr OrMoreAhr Ahru Ahru Ahru Ahru Ahru Ahru Ahru	a Sentence 00142 a Sentence 00058 a Sentence 00055 a Sentence 00052 a Sentence 00020 a Sentence 00120 a Sentence 00020 a Sentence 00023 a Sentence 00023	External Adrog - Any source of information provided from a system external to the AHRS inertial system or function. An AHRS system may have multiple modes of operation. As abrown Figure 11: the AHRS system may consist of 1: 21 Attudes and Heading Reference Unit (AHRU). The AHRS turn and site performance requirements are provided in Section 22.5. Different soccuracy coegosites have been specified birle birl of addinguality afMRS of different capabilities. One or more AHRS may be used on an arcent to supply attude and heading information to the picts of the arcent to ad them with the control and guidance of the arcent. AHRU - Astude and Heading Reference Unit. This document consides an expanient configuration consisting of an Attude and Heading Reference Unit (AHRU) with complementary peripheral equipment specified by the AHRS manufacturer. For example, the "expanient" may include an AHRU. In associated mounting targ, and a magnetic sensing unit.
where a column header here to group by the main truck-ford Heading Reference System ttude-And Heading Reference Unit ttude-And Heading Reference Unit	Btm_Md_Label ecopys	AhrsinetialSystem AhrsSystem AhrsSystem AhrsTum DetrgushingAhr OrdhoreAhr Ahru Ahru Ahru Ahru Ahru Ahru Ahru A	a Sentence 00142 a Sentence 00068 a Sentence 00028 a Sentence 00052 a Sentence 000120 a Sentence 00020 a Sentence 00023 a Sentence 00023 a Sentence 00028	Edemai Adrog - Any source of information provided from a system external to the AHRS inertial system or function. An AHRS system may have multiple modes of operation. An AHRS system may have multiple modes of operation. An AHRS turn and sip performance requirements are provided in Section 2.2.5. Different accuracy categories have been specified to help in distinguishing AHRS of difference but (AHRU). The AHRS turn and sip performance requirements are provided in Section 2.2.5. Different accuracy categories have been specified to help in distinguishing AHRS of different capabilities. One or more AHRS may be used on an arcent to subply attice and heading information to the pilots of the aircraft to add them with the control and guidance of the aircraft. AHRU -Aittude and Heading Reference Unit. This docurrent condition and operating of an AttRUst and Heading Reference Unit (AHRU) with complementary perpileral equipment specified by the AHRS manufacturer. For example, the "equipment" may include an AHRU, an associated mourting tary, and a magnetic sensing unit. As shown in Figure 1-1, the AHRS system may consist of : 1.2 Attude and Heading Reference Unit (AHRU). The AHRU houses. Instelli sensing unit which includes as sension accelerance area and/or inclinometers.
Carg a column header here to group by the mit and theoding Reference System tittude And Heading Reference Lint tittude And Heading Reference Lint	Bm_Mid_label ccopys	AhrsinetialSystem AhrsSystem AhrsSystem AhrsDyn DistrguishingAhr OrMoreAhr Ahru Ahru Ahru Ahru Ahru Ahru Ahru A	a Sentence 00142 a Sentence 00058 a Sentence 00058 a Sentence 00055 a Sentence 00052 a Sentence 00120 a Sentence 00020 a Sentence 00023 a Sentence 00023 a Sentence 00029 a Sentence 00029	Edenal Adrog - Any source of information provided from a system external to the AHRS inertial system or function. An AHRS system may have multiple modes of operation. As a horn in Figure 11: the AHRS system may constat of 1: 21 Attude and Heading Reference Unit (AHRU). The AHRS turn and allo performance requirements are provided in Section 22.5. Different accuracy composites have been pended to help in definition of the Adrog Reference Unit (AHRU) with complementary peripheral equipment specified by the AHRS manufacturer. For example, the "equipment" may include an AHRU, an associated mounting tray, and a magnetic beening unit. For example, the "equipment" may include an AHRU, an associated mounting tray, and a magnetic beening unit. For example, the "equipment" may include an AHRU, an associated mounting tray, and a magnetic beening unit. For example, the "equipment" may include an AHRU, an associated mounting tray, and a magnetic beening unit. The AhrUn Includes. Instal earing unit which includes a series of angle rate series and/or (AHRU). The AHRU includes. Instal earing unit which includes a series of angle rate series and/or (AHRU). The AHRU includes. The AHRS synchrological on constating of an ARtude and Heading Reference Unit (AHRU). The AHRU includes. Instal earing unit which includes a series of angle rate series and/or (AHRU).
In a column header here to group by the main trade-off-backgroup by the third-fold-backgroup by the stude-hold-backgroup by the stude-hold-backgroup between System third-hold-backgroup Between System the System the System third-hold-backgroup Between System third-hold-backgroup Between System the System	Bm. Mid_Label «copy»	Arshvetal System Ars System Ars System Ars System Distruguehrng Pr Onkoe Ar Arsu Arsu Arsu Arsu Arsu Arsu Arsu Ar	aSentence 00142 aSentence 00068 aSentence 00028 aSentence 00052 aSentence 00052 aSentence 00020 aSentence 00020 aSentence 00023 aSentence 00023 aSentence 00023 aSentence 00023 aSentence 00023 aSentence 00023	External Adrig - Any source of information provided from a system external to the AHRS inertial system or function. An AHRS system may have multiple modes of operation. A a HHRS system may have multiple modes of operation. A a HHRS turn and sho performance requirements are provided in Section 2.2.5. Different accuracy categories have been sported to help in disruptionally AHRS of difference but (AHRU). The AHRS turn and sho performance requirements are provided in Section 2.2.5. Different accuracy categories have been sported to help in disruptionality of different capabilities. One or more AHRS may be used on an anothit to acply situate and heading Reference Unit (AHRU) with complementary perpheral equipment specified by the AHRS manufacturer. For example, the "equipment" may include an AHRU an associated mounting train, and a magnetic sensing unit. a shrown in Figure 1.1 the AHRS System may consist of 1.2.1 attuites and Heading Reference Unit (AHRU) with complementary perpheral equipment specified by the AHRS manufacturer. For example, the "equipment" may include an AHRU, an associated mounting train, and a magnetic sensing unit. AHRU includes: herital sensing unit which includes a series of angular rate sensors, accelerometers and/or inclinometers. This document considers an equipment configuration consisting of an Attude and Heading Reference Unit (HHRU) with complementary perpheral equipment specified by the AHRS manufacturer. This AHRU includes: herital sensing unit which includes a series of angular rate sensors, accelerometers and/or inclinometers. This document considers an equipment configuration consisting of an Attude and Heading Reference Unit (HHRU) with complementary perpheral equipment specified by the AHRS manufacturer. This AHRU includes: herital sensing unit which includes a series of angular rate sensors, accelerometers and/or inclinometers. This document considers an equipment configuration consisting of an Attude and Heading Reference Unit (HHRU) with complementary perpher
Ang a column header here to group by the minimum of the solar particular system titude And Heading Reference System titude And Heading Reference System titude And Heading Reference System titude And Heading Reference Lint titude And Heading Reference Lint	Btm_Mid_Label «copy» «copy»	Anshenda System Ansisytem Ansisytem Ansisytem Ansisytem Ansi Ansi Ansi Ansi Ansi Ansi Ansi Ansi	aSentence 00142 aSentence 00068 aSentence 00028 aSentence 00028 aSentence 00120 aSentence 00120 aSentence 00120 aSentence 00028 aSentence 00028 aSentence 00029 aSentence 00029 aSentence 00072	External Adrig - Any source of information provided from a system external to the AHRS inertial system or function. An AHRS system may have multiple modes of operation. An AHRS turn and to performance requirements are provided in Sector 2.25. Different accuracy collegistic colle
Intel Action of the set of the se	Btm_Mid_Label ccopys	Arishetal System Aris System Aris System Aris System Aris System Aris System Aris System Aris Aris Aris Aris Aris Aris Aris Aris	a Sentence 00142 a Sentence 00068 a Sentence 00058 a Sentence 00055 a Sentence 00052 a Sentence 00120 a Sentence 00120 a Sentence 00123 a Sentence 00029 a Sentence 00029 a Sentence 00029 a Sentence 00072 a Sentence 00072 a Sentence 00072 a Sentence 00072 a Sentence 00072	Edemai Adrig - Any source of information provided from a system external to the AHRS inertial system or function. An AHRS system may have multiple modes of operation. An AHRS system may have multiple modes of operation. An AHRS turn and aligo performance requirements are provided in Section 2.2.5. Different accuracy categories have been specified to help in distinguishing AHRS of difference Unit (AHRU). The AHRS turn and aligo performance requirements are provided in Section 2.2.5. Different accuracy categories have been specified to help in distinguishing AHRS of difference Unit (AHRU). The AHRS turn and aligo performance requirements are provided in Section 2.2.5. Different accuracy categories have been specified to help in distinguishing AHRS of difference Unit (AHRU). The AHRS turn and aligo performance requirements are provided in Section 2.2.5. Different accuracy categories have been specified to help in distinguishing AHRS of difference Unit (AHRU). The AHRI considers an equirement configuration consisting of an Attitude and Heading Reference Unit (AHRU) with complementary perpheral equipment specified by the AHRS manufacturer. For example, the "equipment" may include an AHRU, an essociated mounting tray, and a magnetic sensing unit. As shown in Figure 1-1, the AHRS system may consist of 1.2.1. Attude and Heading Reference Unit (AHRU). The AHRU includes: Intel is intelling to include and consisting of an Attitude and Heading Reference Unit (AHRU). The AHRI Chackesyn Simg Semples. 1.7.2. Category Strings Second coperating mode has to sun category drine. 1.7.2. Category Strings Second
Carl a column header here to group by the met that dehading Reference System that dehading Reference Lint that dehading Reference Lint at ego y String at ego y String at ego y String at ego y String ta	Btm_Mid_Label «copy» «copy»	Arshvetal System Ars System Ars System Ars System Dating Jahrny Arr Orklose Arr Anu Anu Anu Anu Anu Anu Anu Anu Category String Example Category String Example Category String Stample	aSentence 00142 aSentence 00058 aSentence 00058 aSentence 00058 aSentence 00058 aSentence 00028 aSentence 00020 aSentence 00020 aSentence 00028 aSentence 00028 aSentence 00029 aSentence 00079 aSentence 00079	Edemail Ading - Any source of information provided from a system external to the AHRS inertial system or function. An AHRS system may have multiple modes of operation. An AHRS system may have multiple modes of operation. An AHRS turn and the performance requirements are provided in Section 22.5. Different sourcessory collegistic additional additinadditional additional additional additional additinad
Intel Action of the set of the se	Btm_Mid_Label «copy»	Arishetal System Aris System Aris System Aris System Aris System Aris System Aris System Aris Aris Aris Aris Aris Aris Aris Aris	a Sentence 00142 a Sentence 00068 a Sentence 00058 a Sentence 00055 a Sentence 00052 a Sentence 00120 a Sentence 00120 a Sentence 00123 a Sentence 00029 a Sentence 00029 a Sentence 00029 a Sentence 00072 a Sentence 00072 a Sentence 00072 a Sentence 00072 a Sentence 00072	Edemai Adrig - Any source of information provided from a system external to the AHRS inertial system or function. An AHRS system may have multiple modes of operation. An AHRS system may have multiple modes of operation. An AHRS system may have multiple modes of operation. An AHRS turn and aligo performance requirements are provided in Section 2.2.5. Different accuracy categories have been specified to help in distinguishing AHRS of different capabilities. One or more AHRS may be used on a moral to subply attribute and heading Reference Unit (AHRU). The AHRS turn and aligo performance requirements are provided in Section 2.2.5. Different accuracy categories have been specified to help in distinguishing AHRS of different capabilities. One or more AHRS may be used on a moral to subply attribute and heading Reference Unit (AHRU), with complementary perpiheral equipment specified by the AHRS manufacturer. For example, the "equipment" may include an AHRU, an essociated mounting tray, and a magnetic sensing unit. As shown in Figure 1-1, the AHRS system may consist of 1.2.1 Attude and Heading Reference Unit (AHRU). The AHRU houses. Instell ensing unit which includes as acceled on stems and/or inclinometers. The AHRU houses. Instell ensing unit which includes are acted and heading Reference Unit (AHRU). The AHRE Chactegory Simp Samples. 1.7.2 Category Simps Samples. 1.7.2

Figure 5 – output of copyDependenciesWithSentences script.

These scripts were written to support the work with the ontology:

- Attributes lists all attributes in the model containing a given string (passed to the script as the Search Term).
- **ClassesExceptOff** for a given string lists all matching names of classes in the model with the exception of the classes contained in the "Off" package.
- **Connectors** lists all associations in the model whose name matches the given string.
- findPackageByName lists package_ID for all packages with the given name (the model can contain packages with the same name and sometime it is necessary to point to a specific one via the package_ID)
- classesInSubpackages lists all the classes contained in the package given by its package_ID or in any of its subpackages.
- **qualifiedClassesInSubpackages** dtto, + qualified class name (e.g. Model.Domain Model.PackageLevel1.PackageLevel2.ClassName).
- mandatoryRequirements lists all requirements whose Status is "Mandatory".
- **copyDependenciesWithSentences** (Lexiana related) for classes in the given package (package_ID) the script lists the superceded "synonyms" of the classes and the sentences, where these superceded terms were present.

Note:



As the SQL implementation does not support transitive closure of the relationships, some scripts are limited in the depth of nesting (classesInSubpackages, qualifiedClassesInSubpackages, copyDependenciesWithSentences).

2.2.2 Automation scripts

The automation (javascript) scripts are primarily intended to automate the work with ontologies that are generated by Lexiana. Some of them might be useful outside of this context, e.g. Lxn-addSuperclassesOfClass, Lxn-hideForeignRelations, Lxn-addNeighboursOfClass, Lxn-underspecifiedRelations. These scripts usually retrieve or modify information about the model.

Example: what the Lxn-1_listCommonReqs script writes in the Output pane is shown in the following figure. In fact, some things contained in the figure are the result of applying several other scripts listed below.

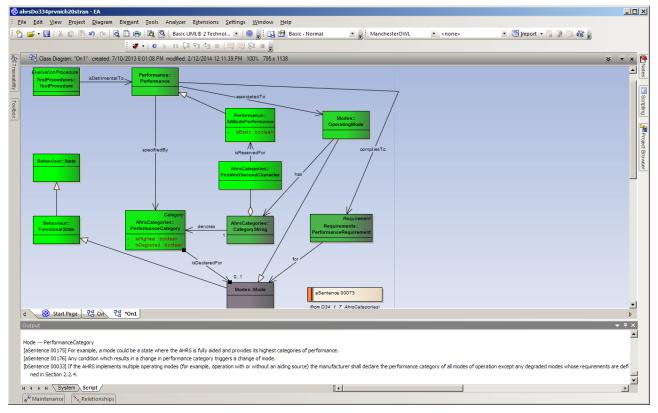


Figure 6 – Work with automation scripts in Enterprise Architect.

- Lxn-1_listCommonReqs (used in diagram) for the selected association writes all requirements which contain both classes related by the association.
- Lxn-2_listCommonReqsDepth (diagram) as above, but considers also all combinations of subclasses of the two associated classes (explosion prevented by the limitation to just the children and grandchildren).
- Lxn-addInhabitants (diagram) for the selected requirement(s) adds all classes to the diagram which Realize the requirement.

Version	Nature	Date	Page
V03.00	R	2014-04-30	13 of 30



- Lxn-addNeighbours (diagram) for the current package adds classes to the diagram, which are related to the classes in the package.
- Lxn-addNeighboursOfClass (diagram) for the selected class(es) adds all classes to the diagram which are related by some kind of association to the selected class.
- Lxn-addPrefix (used in Project Browser) .. changes names of requirements contained in the selected package, e.g. "Sentence 00001" -> "bSentence 00001"
- Lxn-addSuperclassesOfClass (diagram) for the selected class(es) it adds all their superclasses (recursively to the topmost classes) to the diagram.
- Lxn-addTentativeNeighbours (diagram) special case of what Lxn-addNeighbours does.
- Lxn-copyLinks (Project Browser) package selected, for each class in the package, if there is a <<copy>> dependency from the class, the relationships of the class are all copied to the target of the <<copy>> link, then the stereotype <<toBeRemoved>> is added to the source class.
- Lxn-cutOff (Project Browser) for the selected package, for each class in the package each link is removed if it leads to a class contained in an "Off" (trashbin) package.
- Lxn-hideForeignRelations (diagram) hides all relations between classes where neither of the two related classes belongs to the current package.
- Lxn-listReqsOfClass (diagram) for the selected class related requirements are reported in the Output pane.
- Lxn-mirrorReqPackageToObjPackage (Project Browser) creates new packages and distributes the classes into the packages depending on the packaging of requirements within a selected package.
- Lxn-underspecifiedRelations (Project Browser) list unspecified (anonymous) relationships of classes within the selected package and its subpackages.

2.2.3 Traceability view

The Enterprise Architect can provide a lot of different views on the model data. The Traceability view proved to be especially useful while working with the UML ontology. This view provides a similar presentation of the data and relationships between them as can be seen in ontology tools (e.g. Protege). The elements contained in the view can be opened or added to the current diagram.

The diagram in Figure 7 was easily created from the Rotorcraft class by using the Lxn-addSuperclassesOfClass automation script.



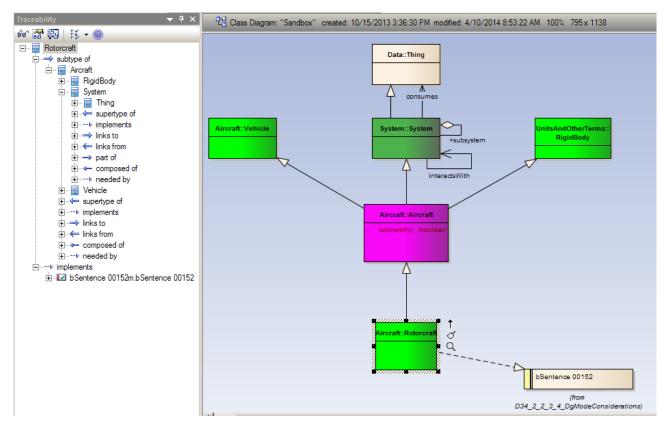


Figure 7 – Traceability view on the left and the corresponding elements in the diagram.

2.2.4 Example

An ontology in the Distributed Ontology Language (DOL) consists of modules formalised in basic ontology languages, such as OWL (based on description logic) or Common Logic (based on first-order logic with some second-order features). These modules are serialised in the existing syntaxes of these languages in order to facilitate reuse of existing ontologies. DOL adds a meta-level on top, which allows for expressing heterogeneous ontologies and links between ontologies. Such links include (heterogeneous) imports and alignments, conservative extensions (important for the study of ontology modules), and theory interpretations (important for reusing proofs). Thus, DOL gives ontology interoperability a formal grounding and makes heterogeneous ontologies and services based on them amenable to automated verification. [Lange, 2012].

Lexiana extracted classes from the text of the paragraph in italics above and generated UML diagrams automatically. This is the diagram with all the extracted classes:



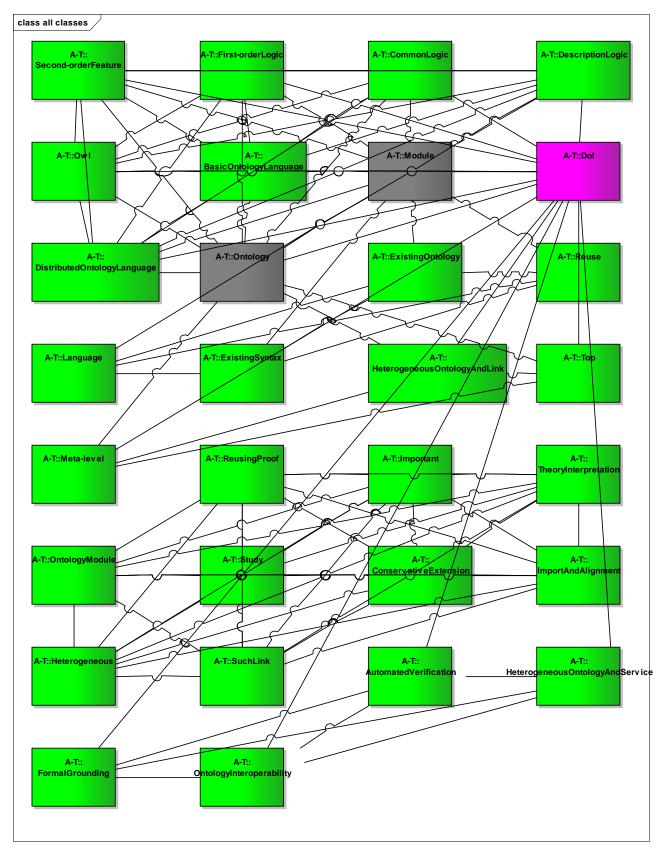


Figure 8 – A diagram generated by Lexiana as shown in the Enterprise Architect.

Version	Nature	Date	Page
V03.00	R	2014-04-30	16 of 30



After manual changes (which took less than 2.5 man-hours) supported by the other generated diagrams and available scripts the UML ontology depicted on the following two figures was obtained:

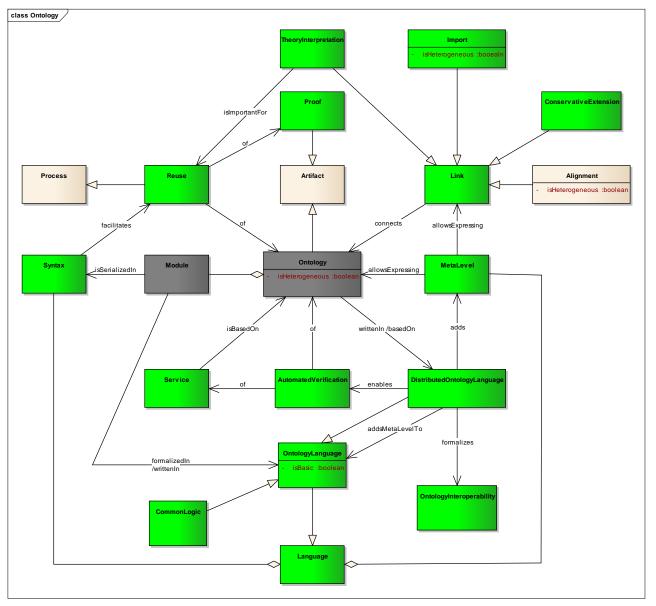


Figure 9 – Resulting UML ontology after manual modifications, part 1.



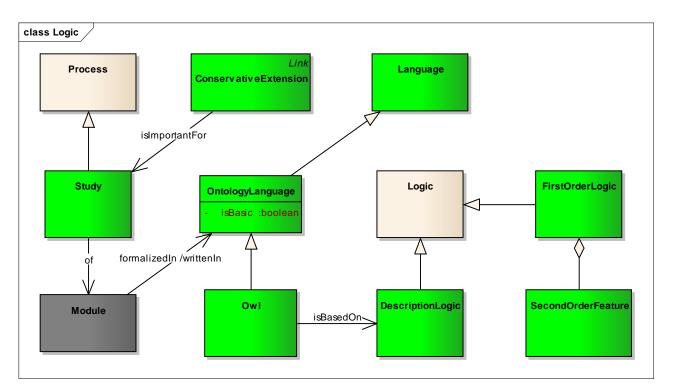


Figure 10 – Resulting UML ontology after manual modifications, part 2.

2.3 knowledgeMANAGER

Thesaurus generated from Enterprise architect using Honeywell scripts will be imported to knowledgeManager. Thesaurus should conform to ISO 25964 and ANSI/NISO Z 39.19:

http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=53657

http://www.niso.org/apps/group_public/download.php/12591/z39-19-2005r2010.pdf

However, there are some differences. For example ANSI/NISO Z 39.19 defines that the following abbreviations BTP (broader term partitive) and NTP (narrower term partitive) **may** denote hierarchical whole-part relationship while knowledgeManager uses WH (whole) and PA (part) abbreviations instead.

Thesaurus file (.the) contains terms and its relationships. For example:

Airplane	
PA: Engine	
PA: Wings	
PA: Navigation System	
Navigation System	
NT: Inertial Navigation System	



List of the thesaurus (.the) file terms and its abbreviations we plan to use:

BT	Broader term (used for generalization)
NT	Narrower term (used for generalization)
PA	Part (used for aggregation)
WH	Whole (used for aggregation)
other	User defined terms

Other terms could be listed in knowledgeManager: Domain Management -> Light Ontology -> Semantics:

💪 knowledgeM#	NAGER								_0>	<u>u</u>
knowledge	MANAGEF	2								
By The	e REUSE Compan	ıy							~	and inform
Domain Mar	agement	Subdom	ains	Indexing & Search	Document	s	Configuration	Save 🧕	🕖 🛛 Exit 🗐	and publis
ight Ontolo	gy		Domain M	anagement > Light Onto	logy > Semantics					
	Tags		≪ ") ("	🠇 🗈 🛍 🖉 🤣	4					
	Semantics		Search fields	Semantic:	Select semantic					_ !
•				hesaurus: 🔽 Identificator:	Semantic configuration:					
					Name:	Generalisa	ation-Specialisation			
				imerate» iivalence»	Ponderation:		1 🛨			
			eExe	mplification»	KM Code:		200 ÷			
				neralisation-Specialisation»	-Normalization properties	s:				
				Hierarchy» Inverse Generalisation-Speciali	Normalized semantic:	«Generalis	sation-Specialisation»			<i>»</i>
				neric Relationship»	Rotate concept:					
			«Hol e «Hol e «Hyp		-Relationship properties					
a Terminology			eldio	matic»	Allow reflexive:		Allow loops:	Allow transitive: 🗖	Symmetrical:	
👂 Tokenizatio	n		● «Link		Thesaurus properties:					
Stemming			🔵 «Mer	onymy»	Show in the thesaurus					
Tagging			«Met @ «Ocr	adata» currence»	Export properties:					
			🗉 🗉 «Orig	gin»	Role A description:	Broader te	em	Role A abbreviation:	BT	
Light Ontola	gy			ographic Equivalence» duct-Material»	Role B description:	Narrower	term	Role B abbreviation:	NT	
Inference ru	les		🔘 «Rea	alization» nantic Structural Association»					Ok	Cancel
👌 Clusterizatio	n								UK I	Cancer
		» *	145 semantics		0 terms					
eady					Connected as	'Admin' to 'C	:\Temp\hratky\rqs\FCU\Rqa	Quality Analyzer v4.1E	inglish_FCU.mdb'	.:

Figure 11 – Thesaurus semantics of the relationships between terms.

Since this process is supposed to be performed only once per domain and by ontology experts only, hence there is no need for tight integration like in development or verification process.

There are several technical issues, which we will have to solve together with REUSE Company in order to get integrated tool chain. Some of the issues were already solved and the affected tools were fixed during our cooperation within Crystal project. The main issues are:

1. The knowledgeManager supports repetitive rules for defining requirement patterns only and grammatical recursion rules cannot be used. Therefore we cannot capture requirement pattern in the form for example: P :== not P | P and P | P or P | term.

Version	Nature	Date	Page
V03.00	R	2014-04-30	19 of 30



This is needed for example to define any logical/arithmetical propositions or any general requirement pattern.

- 2. While Honeywell Lexiana tool and the UML ontology support multiple relations between two terms, the knowledgeManager supports just one relation. There are many terms even in AHRS domain, which have multiple relations. For example:
 - a. Algorithm consumes Data vs. Algorithm produces Data
 - b. System is under Condition vs. System detects Condition
 - c. System derives Output vs. System invalidates Output
 - d. System transitions to Mode vs. System implements Mode vs. System operates in Mode
 - e. Equipment requires Performance vs. Equipment provides Performance
 - f. Equipment Manufacturer reads Equipment Design vs. Equipment Manufacturer modifies Equipment Design
 - g. Manufacturer specifies Data vs. Manufacturer invalidates Data
 - h. Time Period starts at Time vs. Time Period ends at Time.

The knowledgeManager will be used to create requirement patterns (boilerplates) suited for each domain. The requirement patterns will be used in the development tool chain described in the next chapter.



3 Architecture of the Development Tool Chain

This Chapter describes the envisioned architecture of the development tool chain. The engineering methods and tools are depicted in Figure 12 in green boxes. Input and output artefacts are denoted in blue boxes.

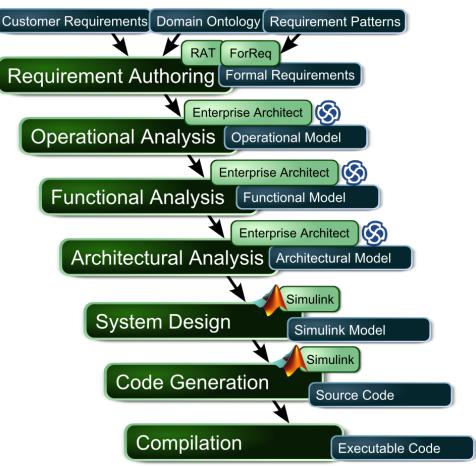


Figure 12 – Development and verification tool chains together.

There will be only small change in development process we propose: requirement authoring will be done using domain ontology and formal requirements will be created in order to enable automatic formal verification and to remove requirement ambiguity.

From the ontology engineering tool chain we expect to get domain ontology and requirement patterns. Afterwards, Requirement Authoring Tool (RAT) can be used to write requirements based on the requirement patterns. ForReq tool (described in Crystal Deliverable D206.010) will formalize the requirements and run automatic verification as described in chapter 4.

Since we provide formal verification of behavioural requirement (functional requirements which define input-output behaviour of the system), hence we need to know which requirement pattern was used to write these requirements. The reason why we need to know which requirement pattern was used is that the behavioural requirement needs to be

Version	Nature	Date	Page
V03.00	R	2014-04-30	21 of 30



mapped to some temporal logic (for example Linear Temporal Logic) so that the requirement can be automatically formally verified.

There are a few options how to achieve the mapping:

- 1. Improve RAT tool with the functionality that it can export both requirement and its requirement pattern (for at least certain types of requirement patterns).
- 2. Extend ForReq tool to enable context-aware auto-completion of requirements based on requirements patterns so that it can substitute RAT functionality completely.
- 3. Improve ForReq tool with the requirement pattern recognition system. Since some requirements might be written based on multiple different requirement patterns, hence this might not be possible at all.

Moreover, in the system design phase we need to map the formal requirement to the design artefacts (variables, states, etc.) so that it can be automatically verified. There are two options how to achieve this mapping:

- 1. Export design artefacts (variables, states, etc.) from ForReq to extend current domain ontology so that RAT can guide the user to use actual system entities while writing requirements. The problem with this approach is that knowledgeManager does not currently support recursive definition in requirement patterns.
- 2. Extend ForReq tool to enable context-aware auto-completion of requirements based on requirements patterns so that it can substitute RAT functionality completely.

When the formal requirements are derived the development process continues with Honeywell Three View System Engineering process which is model based system engineering which created 3 views of the system: operational, functional and architectural model.

At the end, the system design is created in the form of Simulink model and executable code is automatically generated using Honeywell Autocode Manager and Simulink Real-Time Workshop.



4 Architecture of Verification Tool Chain

Verification tool chain will verify the formal requirements provided in Requirements Interchange Format (ReqIF) and system design in Simulink model format.

Since Open Services for Lifecycle Collaboration (OSLC) was selected as a cornerstone technology for tool integration, we are proposing integration architecture in our limited scope and will cooperate with SP6 on Crystal-wide common integration approach.

The proposed OSLC integration technology will enable easy interoperability and exchangeability of the tools if necessary.

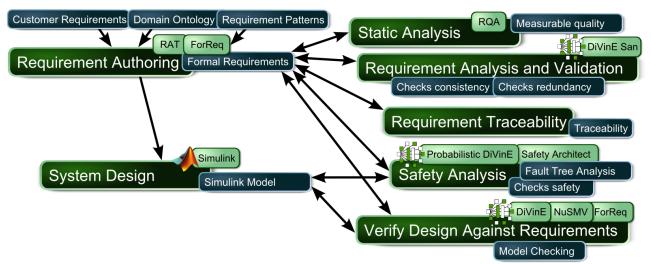


Figure 13 – Verification tool chain architecture with the verification methods.

Static Analysis will be performed by Requirement Quality Analyzer tool from Reuse in order to measure the requirement quality and make sure that the requirements conform to requirement standards. Since RQA is tightly integrated with RAT there will be no need for additional integration effort.

DiVinE Sanity Checking will for all formalized requirements check its sanity, which is composed of two formal verification techniques:

- Consistency checking determines if each requirements subset could be satisfied by some abstract system.
- Redundancy checking determines for each pair of requirements if one requirement does not imply the other. In that case the other requirement is redundant.

Complete description of the sanity checking approach is in [Barnat, 2013].

Requirement Traceability shall be provided by ForReq tool. The formal requirements have direct traces to the system design (in order to enable automatic verification) and also to legacy requirements if any. Also, formal requirements have traces to its formal specification for example in the form of Linear Temporal Logic. Moreover, each verification result will have traces to the requirements and system design.



Safety Analysis will consider the system design extended with an error model and fault injections specification. This extended model will be used to perform minimal cut-set analysis, fault trees creation, and probabilistic analysis. The probabilistic analysis will be done by Probabilistic DiViNE. Verification of the extended model will be used to assess correctness of Fault Detection, Fault Isolation and Recovery (FDIR) subsystems.

Verify Design against Requirements will for each fully formalized and selected requirement call available verification tools (explicit-state model checker DiVinE, symbolic model checker NuSMV, data-symbolic model checker DiVinE, test case generation tool) to automatically verify it on the provided system design. Complete description of the verification method is in [Barnat, 2012].

4.1 Automated Verification

The automated verification tasks will be executed on automation servers in parallel which will be integrated using OSLC with the ForReq tool. Therefore, multiple verification tasks will be executed in parallel on multiple automation servers. This will greatly speed up the verification process. The only verification task that cannot be automated is static analysis using RQA. The architecture should support also multiple ForReq client tools to perform independently and compete for the resources of available automation servers.

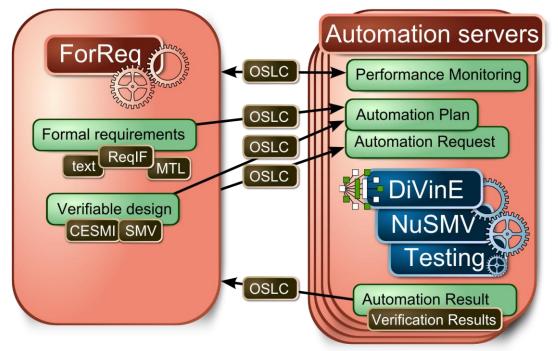


Figure 14 – Integration of ForReq tool and automation servers based on OSLC.

4.1.1 Automation Plan, Request and Result

The ForReq tool will for each verification task that can be automated create an OSLC automation plan and request and send it to best available automation server. Than the verification tools executes the automation plan and sends back the automation results which are reported to the user.

Version	Nature	Date	Page
V03.00	R	2014-04-30	24 of 30



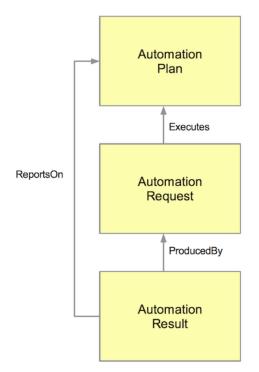


Figure 15 – Relationships among of the OSLC automation specification resources.

Complete specification of the OSLC automation can be found at:

http://open-services.net/wiki/automation/OSLC-Automation-Specification-Version-2.0/

4.1.2 Automation Server Performance Monitoring and Selection

The ForReq tool will have a database of automation servers and will for each verification task select the best automation server. It will choose the server with the most unused CPU and real Memory based on the estimation needs for given verification task, from servers which have the verification tool ready.

To enable this functionality, ForReq will monitor the automation servers using OSLC Performance Monitoring Specification:

http://open-services.net/wiki/performance-monitoring/OSLC-Performance-Monitoring-Specification-Version-2.0

For example this OSLC Resource: Performance Monitoring Record will notifies ForReq that the verification server has 3152 MB of available memory and more than 7 available CPUs:

	<rdf:rdf 02="" 1999="" 22-rdf-syntax-ns#"<="" http:="" th="" www.w3.org="" xmlns:rdf="http://www.w3.or</th><th colspan=4>RDF xmlns:rdf="></rdf:rdf>					
	<pre>xmlns:rdfs="http://www.w3.or</pre>	<pre>xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"</pre>				
	<pre>xmlns:owl="http://www.w3.org</pre>	/2002/07/owl#"				
	<pre>xmlns:dcterms="http://purl.o</pre>	<pre>xmlns:dcterms="http://purl.org/dc/terms/"</pre>				
	<pre>xmlns:rddl="http://www.rddl.</pre>	<pre>xmlns:rddl="http://www.rddl.org/"</pre>				
	xmlns:qudt="http://qudt.org/1.1/schema/qudt"					
Vers	Version Nature Date Page					
V03.00 R 2014-04-30 25			25 of 30			



<pre>xmlns:pm="http://open-services.net/ns/perfmon#"</pre>
<pre>xmlns:ems="http://open-services.net/ns/ems#"></pre>
<rdf:description rdf:about="http://server.address/perf#mem"></rdf:description>
<rdf:type rdf:resource="http://open-services.net/ns/ems#Measure"></rdf:type>
<pre><dcterms:title>Real Free Memory</dcterms:title></pre>
<ems:metric rdf:resource="pm:RealMemoryUsed"></ems:metric>
<ems:unitofmeasure rdf:resource="dbp:MegaByte"></ems:unitofmeasure>
<ems:numericvalue rdf:datatype="http://www.w3.org/2001/XMLSchema#integer"></ems:numericvalue>
3152
<rdf:description rdf:about="http://server.address/perf#cpuu"></rdf:description>
<ems:numericvalue rdf:datatype="http://www.w3.org/2001/XMLSchema#double">157.34</ems:numericvalue
<ems:unitofmeasure rdf:resource="dbp:Percentage"></ems:unitofmeasure>
<ems:metric rdf:resource="http://open-services.net/ns/perfmon#CpuUsed"></ems:metric>
<pre><dcterms:title>CPU Utilization</dcterms:title></pre>
<rdf:type rdf:resource="http://open-services.net/ns/ems#Measure"></rdf:type>
<rdf:description rdf:about="http://server.address/perf#cpus"></rdf:description>
<ems:numericvalue rdf:datatype="http://www.w3.org/2001/XMLSchema#double">8</ems:numericvalue
<ems:unitofmeasure <="" rdf:resource="dbp:quantity" td=""></ems:unitofmeasure>
<ems:metric rdf:resource="http://open-services.net/ns/perfmon#Cpus"></ems:metric>
<dcterms:title>Number of CPU</dcterms:title>
<rdf:type rdf:resource="http://open-services.net/ns/ems#Measure"></rdf:type>

</rdf:RDF>



5 Conclusion

We have described the architecture of the tool chain divided into 3 logical parts. We have shortly described functions of individual technology bricks which we plan to integrate into the tool chain. We have concentrated on inputs and outputs of each technology brick in order to ensure robust tool integration.

We have proposed a few possible options for the requirement authoring architecture. The final approach will depend on the results of our cooperation with REUSE Company. We are very demanding on the functionality of their requirements technology bricks in order to be sure that we can use the full potential of domain ontology engineering.

We have proposed tool integration based on Open Services for Lifecycle Collaboration (OSLC) as a selected cornerstone technology for tool integration. A tight cooperation with the technology bricks providers is a must order to achieve the fully functional tool chain.



6 Terms, Abbreviations and Definitions

3VSE	Three View System Engineering – Honeywell model-based development process
AHRS	Attitude and Heading Reference System
BTP	Broader Term (Partitive)
CTL	Computational Tree Logic
DiVinE	Distribute Verification Environment – model checker from Masaryk University
DODT	Tool that semi-automatically transforms natural language requirements into semi-formal boilerplate requirements using domain ontology.
EA	Enterprise Architect – a tool from SPARX Company
FDIR	Fault Detection, Fault Isolation and Recovery Techniques
ForReq	Formalization of Requirements – internal Honeywell tool
FTA	Fault Tree Analysis
GNSS	Global Navigation Satellite Systems
GUI	Graphical User Interface
IRS	Inertial Reference Systems
kM	knowledgeManager – a tool from REUSE Company
LTL	Linear Temporal Logic
NRP	Narrower Term (Partitive)
OSLC	Open Services for Lifecycle Collaboration
PA	Part
ReqIF	Requirement Interchange Format (adopted as formal specification by OMG)
RAT	Requirement Authoring Tool – a tool from REUSE Company
RQA	Requirement Quality Analyzer – a tool from REUSE Company
San	Sanity checking of requirements (consistency, vacuity and completeness)



SQL	Structured Query Language
SysML	Systems Modelling Language
UML	Unified Modelling Language
WН	Whole
ХМІ	XML Metadata Interchange

Table 6-1: Terms, Abbreviations and Definitions



7 References

[Barnat, 2012]	Tool Chain to Support Automated Formal Verification of Avionics Simulink Designs. J. Barnat and J. Beran and L. Brim and T. Kratochvila and P. Rockai: Formal Methods for Industrial Critical Systems (FMICS 2012), Springer, 2012, volume 7437 of LNCS, 78-92.
[Barnat, 2013]	Checking Sanity of Software Requirements. Barnat, Jiří, Bauch, Petr and Brim, Luboš. Brno: 2013.
[Kotis, 2006]	Towards Automatic Merging of Domain Ontologies: The Hcone-Merge Approach. Konstantinos Kotis, George A. Vouros, Konstantinos Stergiou. Web Semant. 4, 1 (January 2006), 60-79.
[Lange, 2012]	LoLa: A Modular Ontology of Logics, Languages, and Translations. Christoph Lange, Till Mossakowski, and Oliver Kutz: 2012, <u>http://www.informatik.uni-bremen.de/~till/papers/womo2012.pdf</u>