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

Multi-Mode Navigation System Analysis, Development Needs, and the Proposed Tool-Chain Functionality

D206.010

DOCUMENT INFORMATION

Project	CRYSTAL
Grant Agreement No.	ARTEMIS-2012-1-332830
Deliverable Title	Multi-Mode Navigation System Analysis, Development Needs, and the Proposed Tool-Chain Functionality
Deliverable No.	D206.010
Dissemination Level	RE
Nature	R
Document Version	V1.3
Date	2014-01-31
Contact	Jan Beran, Tomáš Kratochvíla
Organization	HON
Phone	+420 532 115 530
E-Mail	Jan.Beran@honeywell.com, Tomas.Kratochvila@honeywell.com

AUTHORS TABLE

Name	Company	E-Mail
Tomáš Kratochvíla	Honeywell	Tomas.Kratochvila@honeywell.com
Jan Beran	Honeywell	Jan.Beran@honeywell.com

REVIEW TABLE

Version	Date	Reviewer
V0.1	Jan 2014	Jan Beran (Internal)
V1.1	Jan 2014	Anne Monceaux (External)
V1.2	Jan 2014	Marc Malot (External)
V1.3	Jan 2014	Approved

CONTENT

Content of Tables

Content of Figures

1 Introduction

1.1 Role of deliverable

This document has the following major purposes:

- Define the overall use case, including a detailed description of the underlying development processes and the set of involved process activities and engineering methods
- Provide input to WP601 (IOS Development) required to derive specific IOS-related requirements
- Provide input to WP602 (Platform Builder) required to derive adequate meta models
- Establish the technology baseline with respect to the use-case, and the expected progress beyond (existing functionalities vs. functionalities that are expected to be developed in CRYSTAL)

1.2 Structure of this document

This document contains **Multi-Mode Navigation System** use case description in chapter 2. The activities needed to accomplish the use case development process and the stakeholders are described in chapter 3.

The engineering methods, including all steps for each activity and its preconditions and post conditions, are detailed in chapter 4. The excel version of the engineering methods, where input and output artefacts are defined, is attached in chapter 7.

2 Use Case Description

The objective of this use case is to instantiate a tool chain for the **Multi-Mode Navigation System** use case, demonstrate functionality of the tool chain, and advance the tool chain to a close-to-production Technology Readiness Level.

Multi-Mode Navigation System **provides information of the aircraft position, velocity, and attitude** to other aircraft systems, such as flight management systems, flight control systems, surveillance systems, as well as to the pilots through the flight displays. We are fusing the navigation systems. Examples of navigation systems:

- **Global Navigation Satellite Systems** (GNSS) – combines some of the existing satellite navigation systems: Galileo, GPS, GLONASS, and Compass with Satellite Based Augmentation System or Ground Based Augmentation System. The planned multi-constellation and multi-frequency GNSS will provide integrity and accuracy monitoring.
- **Attitude Heading Reference Systems** (AHRS) provides attitude, heading and yaw of the aircraft using accelerometers, gyroscopes, and magnetometers on thee axes. On top of inertial reference systems the AHRS also computes the attitude and heading solutions.
- **Inertial Reference Systems** (IRS) provides position, velocity and orientation of the aircraft based on accelerometers and gyroscopes.

The communication of AHRS system with external systems is depicted in Figure 1.

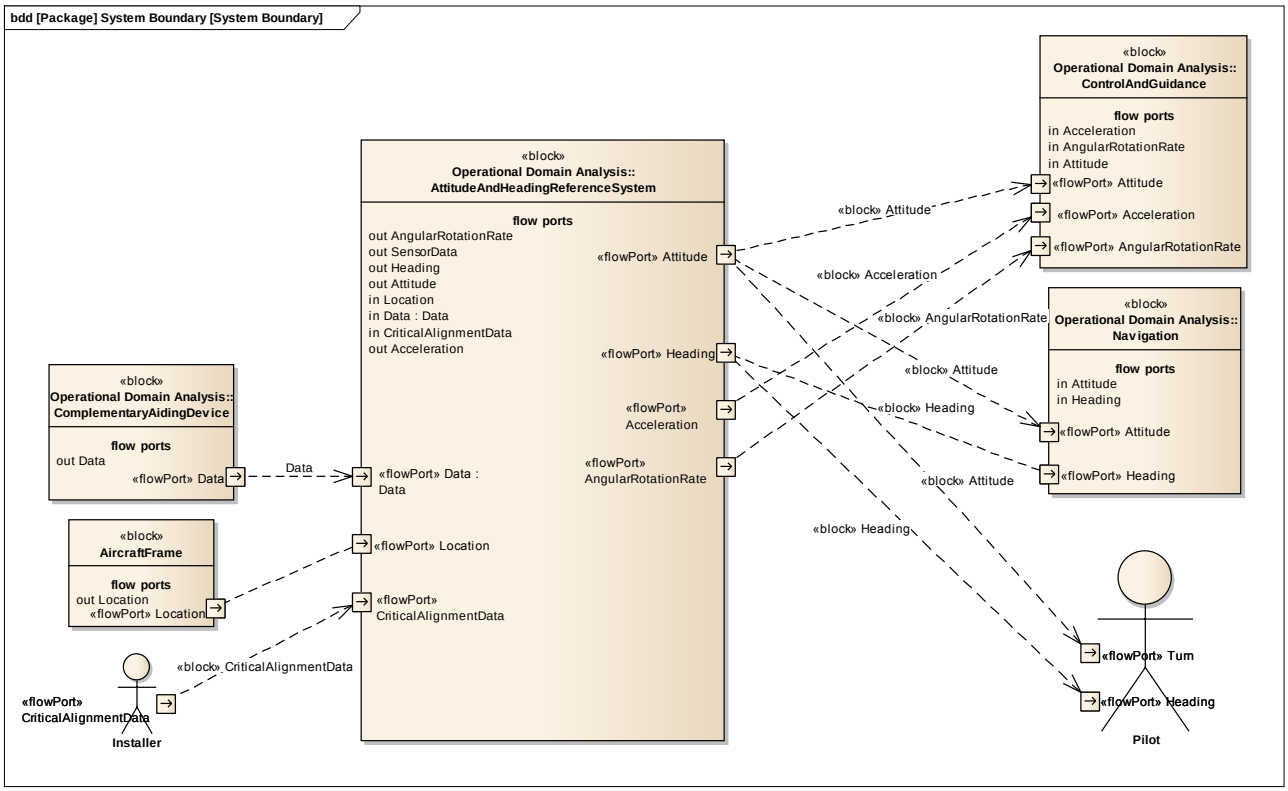

Figure 1 System Boundary diagram sums up the communication of AHRS with external systems

The high-level algorithms used in AHRS together with the inputs and outputs are in Figure 2.

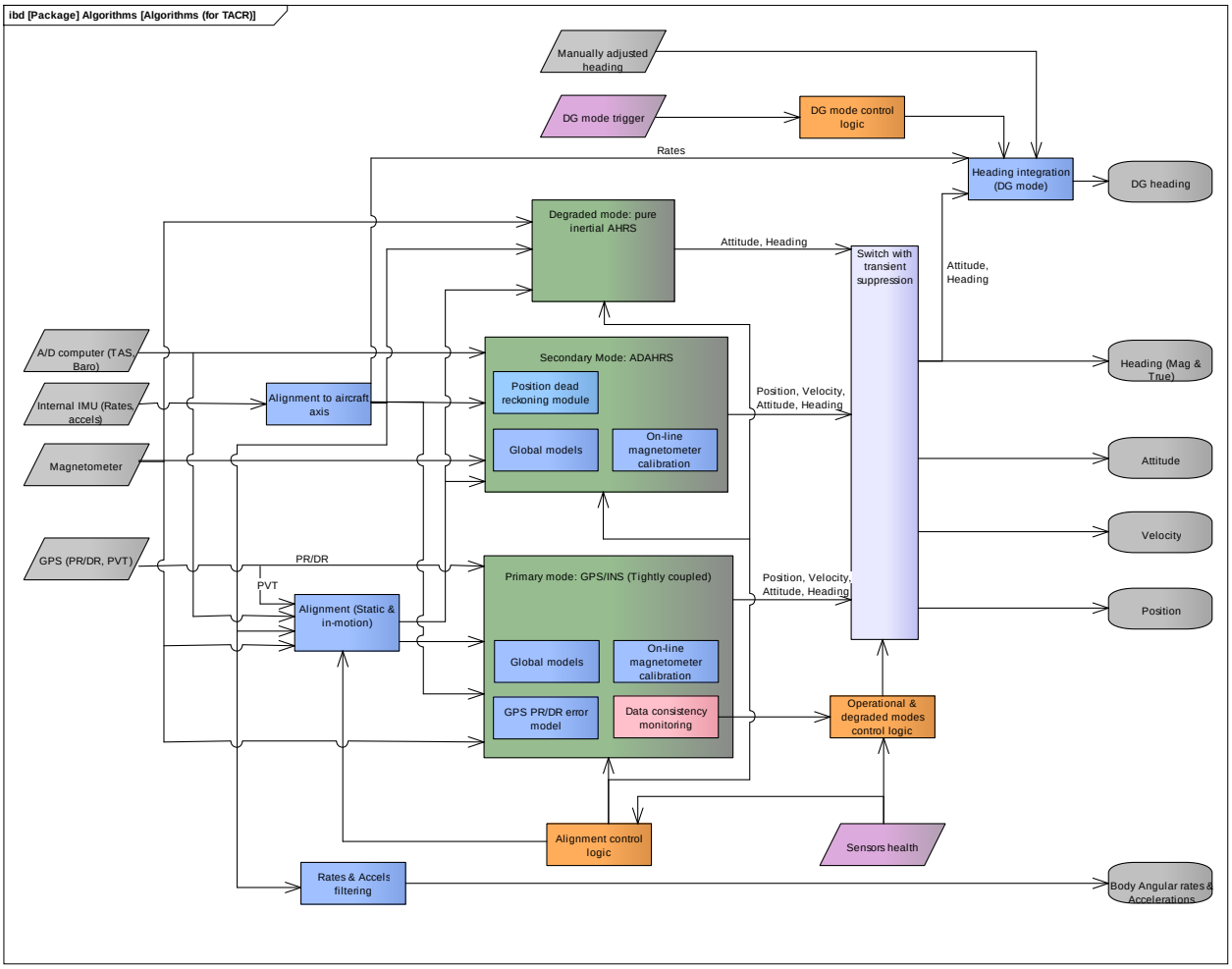

Figure 2 Algorithms used in AHRS

There is a need for trade-off analysis among different architectures of the Multi-Mode Navigation System use case:

- The integrated systems could be tightly or loosely coupled.
- There are multiple integrity monitoring algorithms.
- There is a choice between single or multi constellation of the GNSS.

3 Detailed Description of the Use Case Process

3.1 Activities

The use case development process consists of activities as depicted in Figure 3. The activities do not describe the current state of the art but the envisioned state we want to achieve in Crystal project.

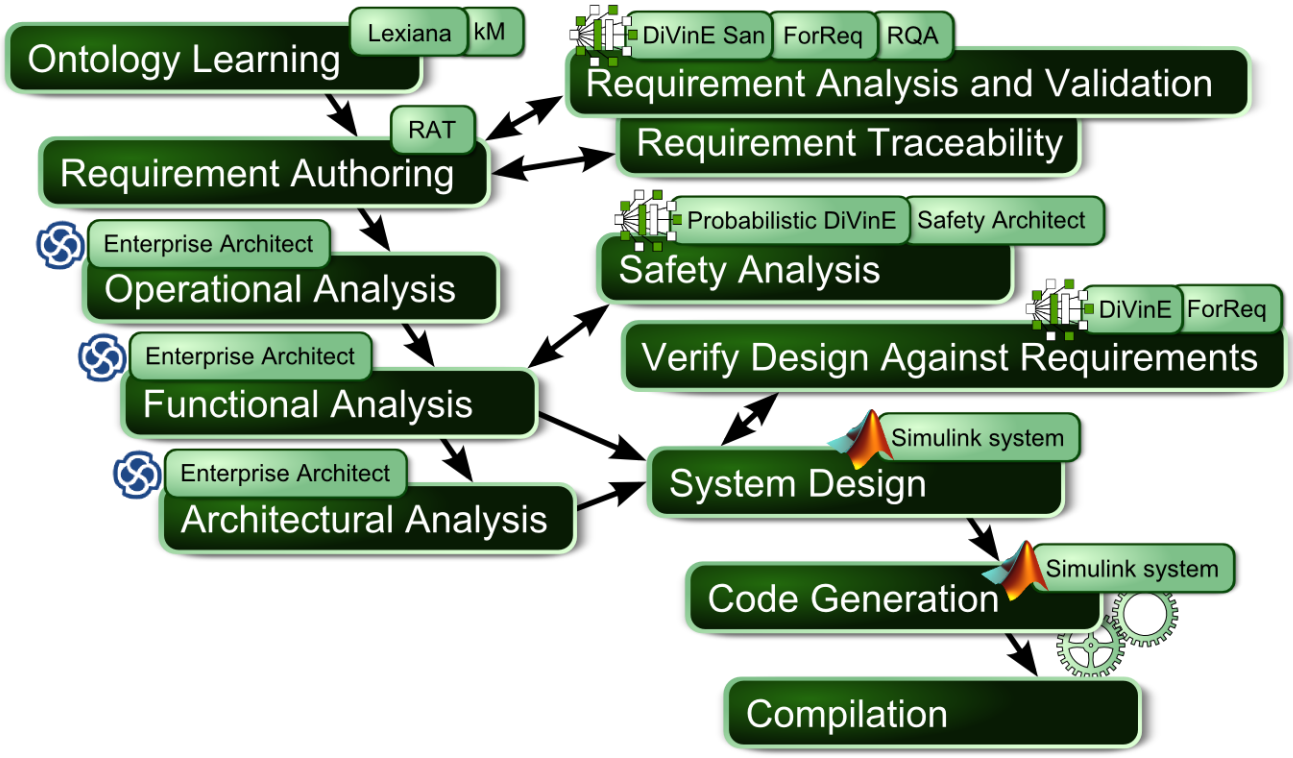

Figure 3 Development process.

Ontology Learning (acquisition, extraction, generation, or mining) activity extracts domain ontology semi-automatically in our case mainly from standards, legislation and customer requirements. One or more ontologies shall be used for all projects within certain area. The ontology will be updated during the complete development process to correspond. The Honeywell *Lexiana* tool, and Enterprise Architect shall be used to create ontology. Afterwards the ontology will be exported to REUSE *Knowledge Manager (kM)* in order to enable its management and further interoperability with other REUSE tools (*Requirements Authoring Tool* and *Requirement Quality Analysis*).

Requirement Authoring captures customer requirements. *Requirements Authoring Tool* from Reuse could be used.

Requirement Analysis and Validation verifies the quality of the requirements. Informal analysis is performed by for example *Requirement Quality Analysis* tool. Formal verification can be performed by *ForReq* tool which formalizes requirements to become unambiguous and machine readable. Afterwards the *DiVinE San* sanity checking tool formally verifies consistency, vacuity

(redundancy) and behavioural completeness of the requirements. More on sanity checking is in paper [Barnat, 2013].

Requirements Traceability is provided by all requirement management tools to ensure that requirements can be traced among all requirement hierarchy levels, versions and associated artefacts (system design blocks, etc.).

Operational Analysis is done using Honeywell Three View System Engineering (3VSE) process using *Enterprise Architect*. Operational analysis of the system is focused on system work that shall be accomplished. It specifically addresses the problem to be solved and does not focus on specific solutions. The analysis captures and clarifies requirements and program expectations from customers and stakeholders and it should consider the operational, development, business and political environments that the system is going to be developed and operated in.

Functional Analysis is done using Honeywell Three View System Engineering (3VSE) process using *Enterprise Architect*. Create the functional view of the system and its requirements that go with functional model. Functional analysis is about what the system must do to accomplish the work defined by the operational model. Thus, it defines the behaviour of the system.

Architectural Analysis is done using Honeywell Three View System Engineering (3VSE) process using *Enterprise Architect* and is focused on the physical model of the system and defines how the system will implement the functional model and requirements. In this analysis we allocate functions and requirements to specific configuration items, i.e. to specific physical parts of the system.

Safety Analysis will be done to using Safety Architect or probabilistic DiVinE. We plan to automate the process, increase its confidence level and make it formal.

System Design is derived from functional design in Mathworks *Simulink* system. The source code is then automatically generated using Mathworks *Simulink* and compiled. Also simulation of the system is done using Mathworks *Simulink.*

Verify Design Against Requirements activity is detailed in Figure 4. It consists of requirement formalization to obtain machine readable requirements and system formalization to capture the semantics of the Simulink system design. Then the *DiVinE* model checker is executed to formally verify whether the given system satisfies the requirements or not. More on the approach is in [Barnat, 2012]

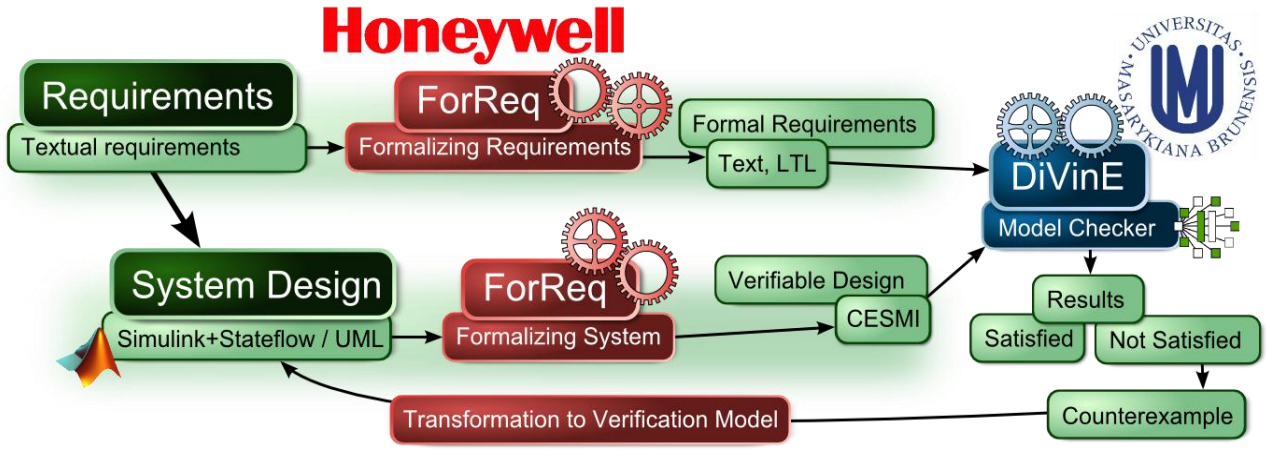

Figure 4 Activities for requirement formalization and formal verification of requirements in detail.

For the following three tools we can alter its interfaces to make sure that the tools can be smoothly integrated with the rest of the proposed tool chain:

DiVinE is a distributed verification environment – an open source explicit-state model checker. Building on high-performance algorithms and data structures, it offers unparalleled versatility, scaling from a typical developer's laptop, up to a high-end compute cluster. What more, it can verify a wide range of languages, including C and C++. More information can be found at: http://divine.fi.muni.cz/. DiVinE will be used as a backend tool for in tool chain to enable requirements validation, design verification against requirements and safety verification. To increase the robustness also NuSVM (http://nusmv.fbk.eu/) symbolic model checker might be used as a backend tool. This approach gets result even if one tool require more processing resources then available, it gets the verification results from the faster tool, and it makes sure the different tools always provide the same results.

ForReq is a Honeywell tool for formalizing requirements and automating verification process (using DiVinE or NuSMV model checkers as a backend). The tool is described in [Barnat, 2012].

Lexiana is a Honeywell tool which creates requirements models from natural text. Such models can be further used in Enterprise Architect to create Domain Ontology.

While system safety assessment process is done in each phase of development process as depicted in Figure 5, we will mainly focus on improving and automating the safety assessment when the functional model is created.

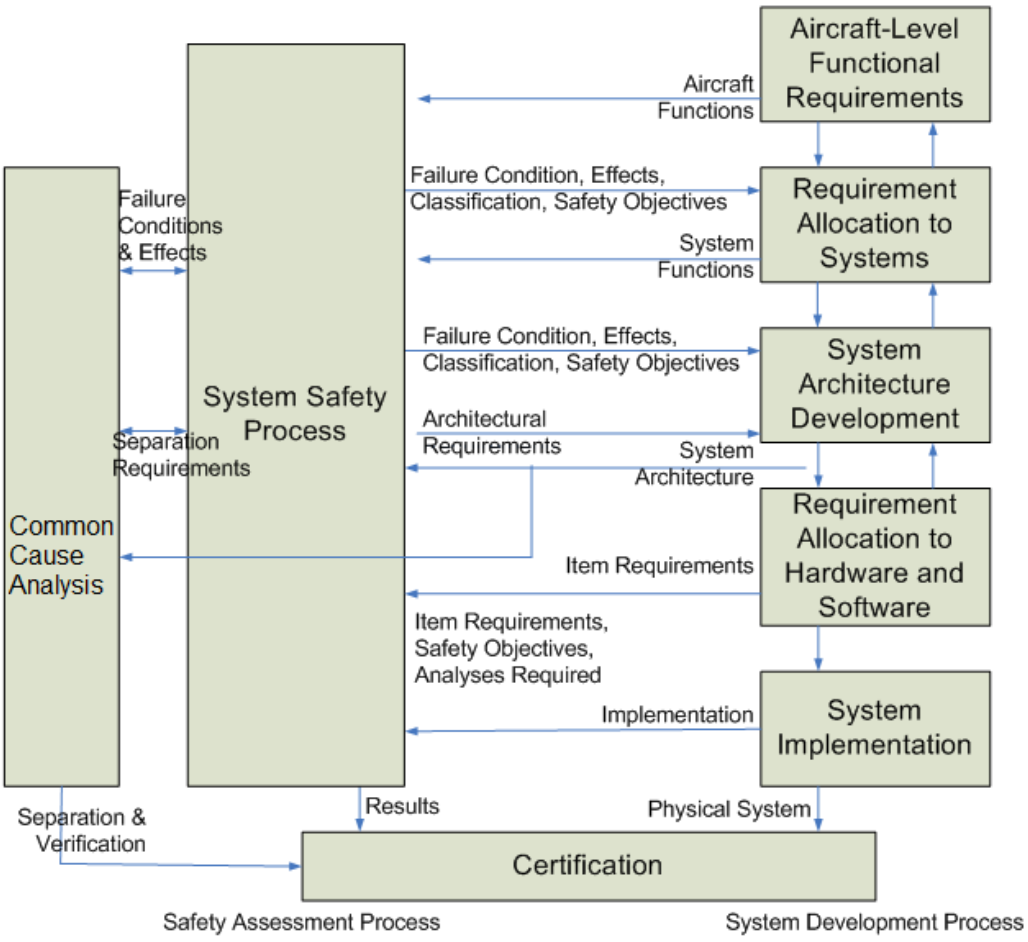

Figure 5 Safety Assessment Process and System Development Process are tightly coupled.

Since Honeywell complies to SAE ARP 4761, RTCA DO-178B/C and related avionics standards, hence the complete process objectives (activities) are much broader as showed in Figure 6. However, we have selected the most important activities which are not automated and we want to improve them to either speed up the development process by automation or increase the quality of the developed system.

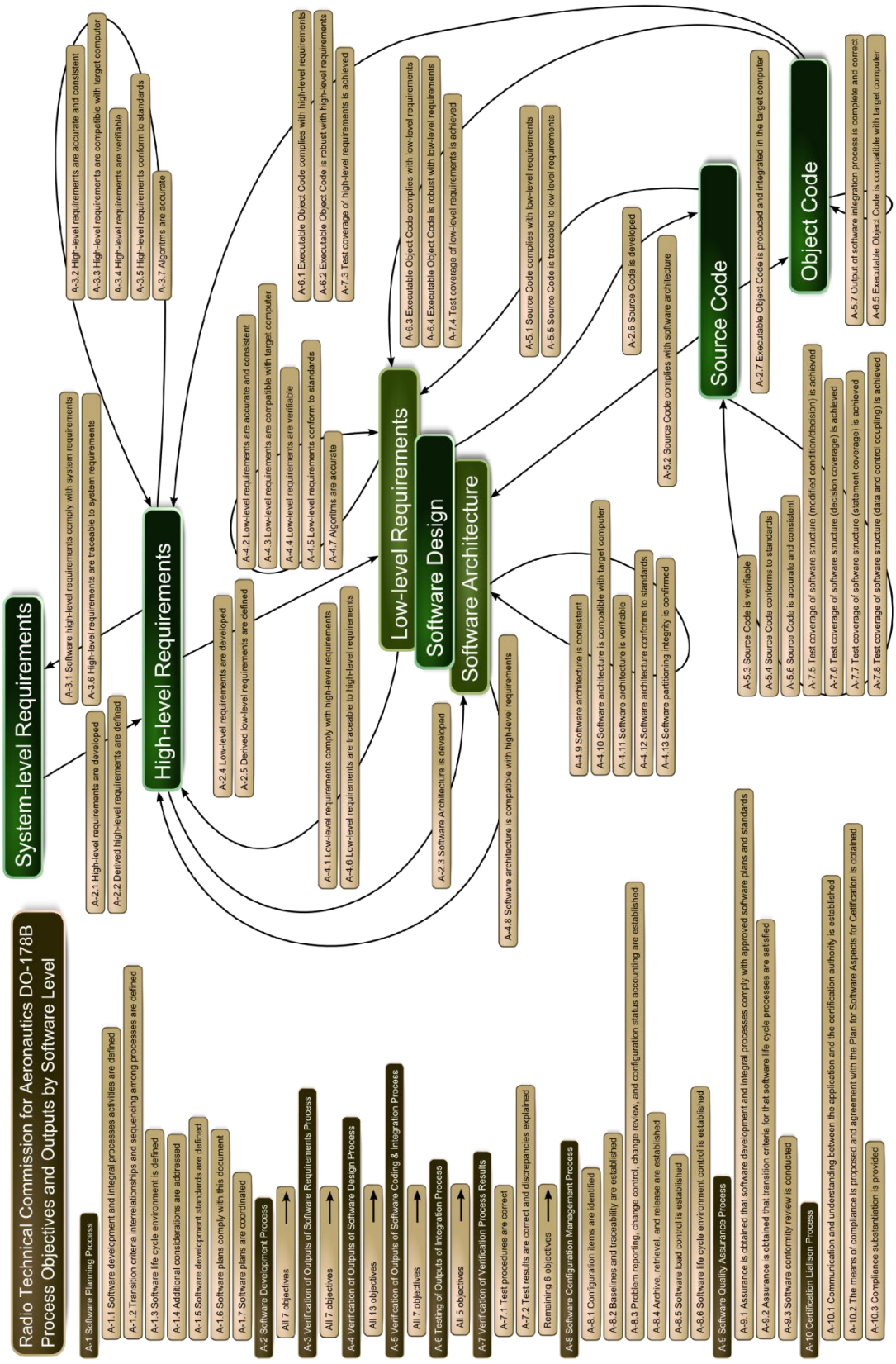

Figure 6 Complete Process Objectives for aerospace domain from DO178B

3.2 Stakeholders & Roles

Stakeholders	Role
Customer	Provides customer requirements
Supplier	Provides products and services
Government	Provides legislation
Standardization Organization	Provides standards
Requirement Engineer	Requirements authoring
Validation Engineer	Requirement analysis
System Engineer	Operational analysis
System Engineer	Functional analysis
System Engineer	Architectural analysis
Verification Engineer	Safety analysis
Software Engineer and Hardware Engineer	System Design
Verification Engineer	Unit Testing
Verification Engineer	System Testing
System Integrator	System Integration

4 Identification of Engineering Methods

The Table 4-1 maps the process activities to engineering methods and involved tools. Tools Involved column list tools we intend to use. We also list alternative tools in (parentheses), which we might use if they perform better. Therefore we will have to evaluate them.

Process Activity	Engineering Method	Tools Involved
Create ontology	Ontology Learning	Lexiana, Enterprise Architect (DODT, Knowledge Manager)
Requirement elicitation	Requirement Authoring	Requirements Authoring Tool
Requirement analysis	Requirement Analysis	RQA
Formalize requirements	Requirement Analysis	ForReq
Sanity checking of requirements	Requirement Sanity Checking	DiVinE San
Trace requirements	Trace Requirements	all tools
Operational Analysis	Operational Analysis	Enterprise Architect
Functional Analysis	Functional Analysis	Enterprise Architect
(Requirements and functional analysis)	(Requirements Temporal Checking)	(RATSY)
Safety Analysis	Fault Tree Generation, Markov Analysis	DiVinE Probabilistic, All4Tec Safety Architect (Windchill FTA)
Create system design	System Design	Simulink (SCADE)
Check system design	Simulations	Simulink (SCADE)
Verify design against requirements	Requirements Verification	ForReq, DiVinE (NuSMV)
Architectural Analysis	Architectural Analysis	Enterprise Architect
Generate source code	Implementation	Simulink
Generate executable	Compilation	Compiler

Table 4-1: Summary of the activities and corresponding engineering methods.

The description of the engineering methods in detail is in the following table. It describes the pre and post conditions for each engineering activity and all the steps which has to be performed.

Engineering Method: Ontology Learning		
Purpose: The method is used to get common understanding of a domain and synchronize terminology. Furthermore, the resulting ontology can be used for analysis and can advise during requirement elicitation.		
Pre condition	Engineering Activity	Post condition

Domain Standards are available. Product line constraints are available. Legislation is available. Customer requirements are available in machine readable format.	1. Read the input documents in Lexiana. 2. In Lexiana the initial ontology is created automatically in the form of Requirements model and saved as XMI. 3. In Enterprise Architect read the initial ontology. 4. In Enterprise Architect the Requirement Engineer clarifies and clusters the terms and its relations. 5. In Enterprise Architect the Requirement Engineer refine relationships and add more information if needed. 6. The ontology is exported to knowledge Manager.	Domain Ontology is available in knowledge Manager.

Engineering Method: Requirement Authoring		
Purpose: To capture requirements		
Pre condition	Engineering Activity	Post condition
Customer requirements are available. Domain Standards are available. Product line constraints are available. Legislation is available. Domain Ontology is available in knowledge Manager.	1. The analyst checks out the customer requirements from a versioning system. The specification can be of a multitude of formats, DOORS, MS Excel, plain text, MS Word. 2. The Requirement Authoring Tool parses the text and reveals inconsistencies with the ontology and proposes alterations leading to consistent artefact. 3. The analyst rewrites the text and is hinted by some mechanism on how to proceed with building the requirement based on the information contained in the ontology.	System high-Level requirements created.

Engineering Method: Requirement Analysis		
Purpose: To increase quality of requirements		
Pre condition	Engineering Activity	Post condition
System High-Level Requirements stored in ReqIF (Requirement Interchange Format).	1. Read the list of requirements. 2. Analyze the requirements in RQA 3. Correct requirements if necessary. 4. Save the list of requirements	System High-Level Requirements stored in ReqIF (Requirement Interchange Format) which is approved for basic quality level.

	back.	

Engineering Method: Requirement Sanity Checking		
Purpose: Remove ambiguity from the requirements by formalizing them. Validate that the formal requirements are sane. Check its consistency, vacuity (redundancy) and behavioural completeness.		
Pre condition	Engineering Activity	Post condition
System High-Level Requirements stored in ReqIF (Requirement Interchange Format).	1. Read the list of requirements. 2. ForReq guides user to formalize the requirements to be unambiguous. 3. ForReq translates requirements into machine readable properties. 4. Perform sanity checking using DiVinE San for all requirements or just clustered groups of requirements. The consistency, redundancy and completeness is checked. 5. Present the validation results. 6. Correct requirements if necessary. 7. Save the list of requirements back.	Formal System High-Level Requirements stored in ReqIF (Requirement Interchange Format) which are sane (non-conflicting and non-redundant).

Engineering Method: Operational Analysis		
Purpose: Identify the system operations, system boundary, operational scenarios and mission scenarios.		
Pre condition	Engineering Activity	Post condition
Domain Ontology is available. Customer requirements are available in machine readable format.	1. Read the list of customer requirements. 2. Create Operational Model in Enterprise Architect using 3 View System Engineering process. 3. Create System Boundary. 4. Create Operational Scenarios. 5. Create Mission Scenarios. 6. Save the Operational Model in Enterprise Architect.	Operational Model created including system boundary, operational scenarios and mission scenarios.

Engineering Method: Functional Analysis		
Purpose: Identify what the system must do to accomplish the work.		
Pre condition	Engineering Activity	Post condition
Formal System High-Level Requirements stored in ReqIF (Requirement Interchange Format).	1. Read the list of high-level requirements. 2. Create Functional Model in Enterprise Architect using 3 View System Engineering process. 3. Identify system functions and create block definition diagram for each function. 4. Identify and describe all operating modes. 5. Define system or physical states.	Functional Model created.

	6. Build state or mode transition diagram. 7. Map states or modes to functions. 8. Create functional flow block diagram. 9. Create data flow block diagram. 10. Build functional hierarchy. 11. Develop algorithm models. 12. Model information structures within the system and the relationships between different types of information. 13. Save the Functional Model in Enterprise Architect.	

Engineering Method: Safety Analysis		
Purpose: To ensure the safety of the system.		
Pre condition	Engineering Activity	Post condition
Functional Model and system design is created.	1. Read the functional model. 2. Analyze the safety of the system in Safety Architect. 3. Perform the safety analysis of the system in Probabilistic DiVinE. 4. Save the corrected functional model back.	Safety analysis is performed.

Engineering Method: System Design		
Purpose: To design the system.		
Pre condition	Engineering Activity	Post condition
System High-Level Requirements stored in ReqIF (Requirement Interchange Format). Functional model developed.	1. Read the list of requirements. 2. Develop the system in the Simulink based on the Functional Model which the functional hierarchy is preserved.	Executable system design is created.

Engineering Method: Requirement Verification		
Purpose: To verify high-level requirements against system design		
Pre condition	Engineering Activity	Post condition
Formal System High-Level Requirements stored in ReqIF. System Design in Simulink (SCADE).	1. Read the list of formal requirements in ForReq tool. 2. ForReq translates requirements into machine readable properties (Linear Temporal Logic or Computational Tree Logic). 3. Assign the system design to the requirement document. 4. ForReq translates system design into an input language of the DiVinE (NuSVM) model checker. 5. ForReq sends verification plan (OSLC) to DiVinE (NuSVM) model checker. 6. ForReq sends verification request (OSLC) to DiVinE (NuSVM) model checker.	Formal verification report – a proof that system des satisfies the requirements.

	7. DiVinE (NuSVM) model checks the system design against requirements. 8. ForReq reads verification results from DiVinE (NuSVM) model checker. 9. Generate the test cases if the model checking of the requirements cannot be performed on the available server or cluster. 10. Present the verification results to the user. 11. Check the counterexample simulation for the requirements which are not satisfied. 12. Fix the requirement or the system design so that all the requirements are satisfied. 13. Sends corrected formal requirements in ReqIF. 14. Save the formal verification report.	

Engineering Method: Architectural Analysis		
Purpose: Defines how the system will implement the functional model and system requirements.		
Pre condition	Engineering Activity	Post condition
Formal System High-Level Requirements stored in ReqIF (Requirement Interchange Format).	1. Read the list of requirements. 2. Create Architectural Model in Enterprise Architect using 3 View System Engineering process. 3. Create Architectural hierarchy diagram. 4. Define Installation location and physical layout. 5. Create architectural flow diagram. 6. Create physical interconnection diagram. 7. Allocate performance requirements, sequence diagrams. 8. Write Interface definition and structure. 9. Create User interface structure and layout. 10. Save the Architectural Model in Enterprise Architect.	Architectural Model is created.

Engineering Method: Code Generation		
Purpose: To generate the source code.		
Pre condition	Engineering Activity	Post condition
Executable system design is created.	1. Generate the source code from Simulink.	Source code is created.

The following three engineering methods we are focused on have more tools involved and therefore the sequence charts of the process flow is depicted:

- Ontology Learning (Figure 7)
- Requirement Sanity Checking (Figure 8)
- Requirement Verification (Figure 9)

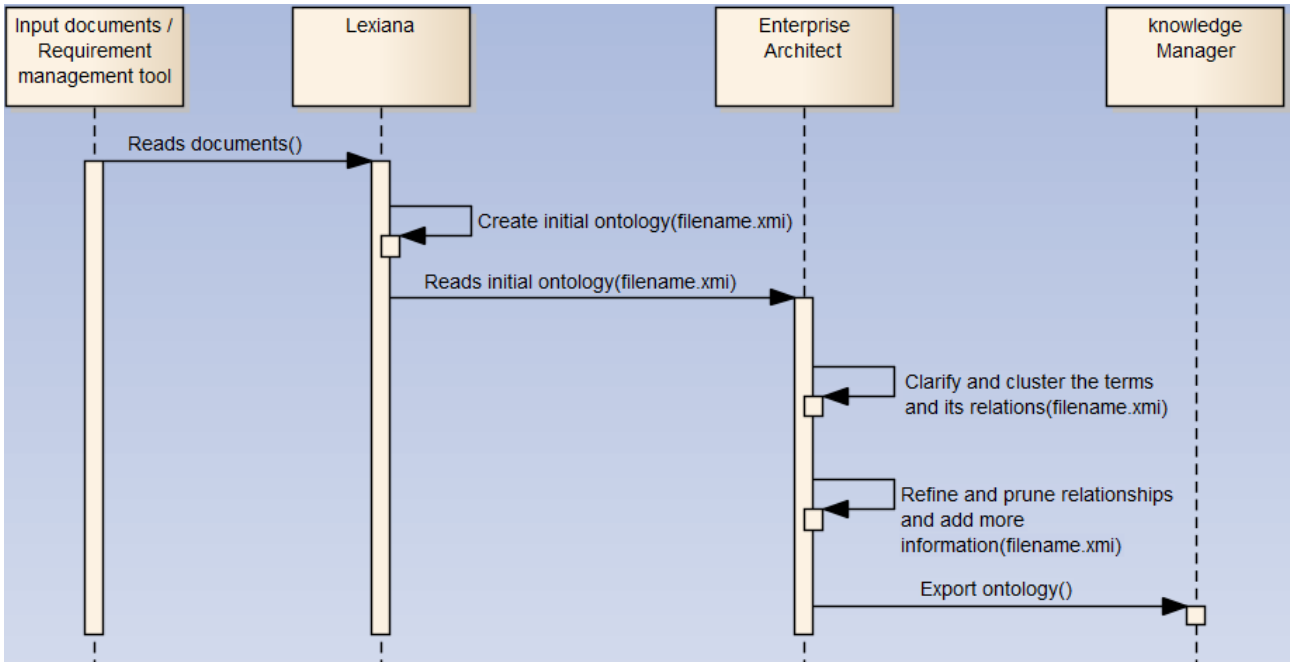

Figure 7 Ontology Learning

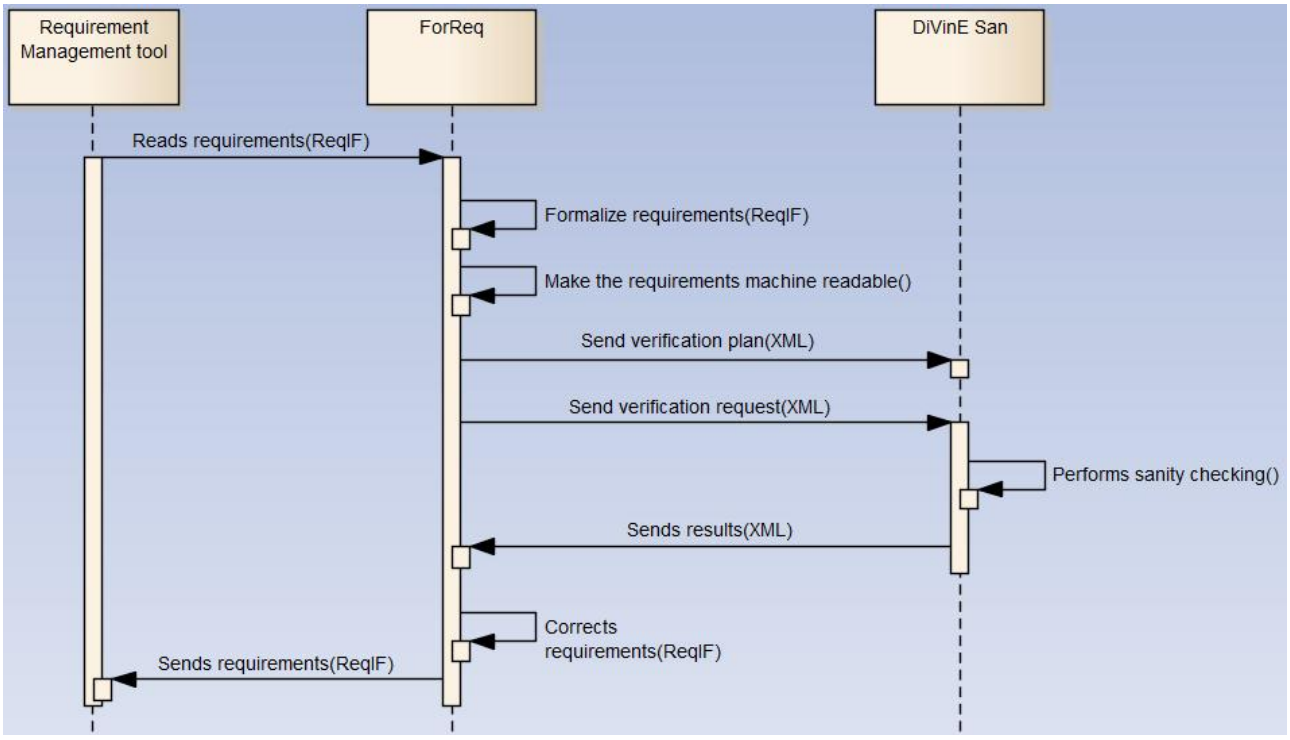

Figure 8 Requirement Sanity Checking

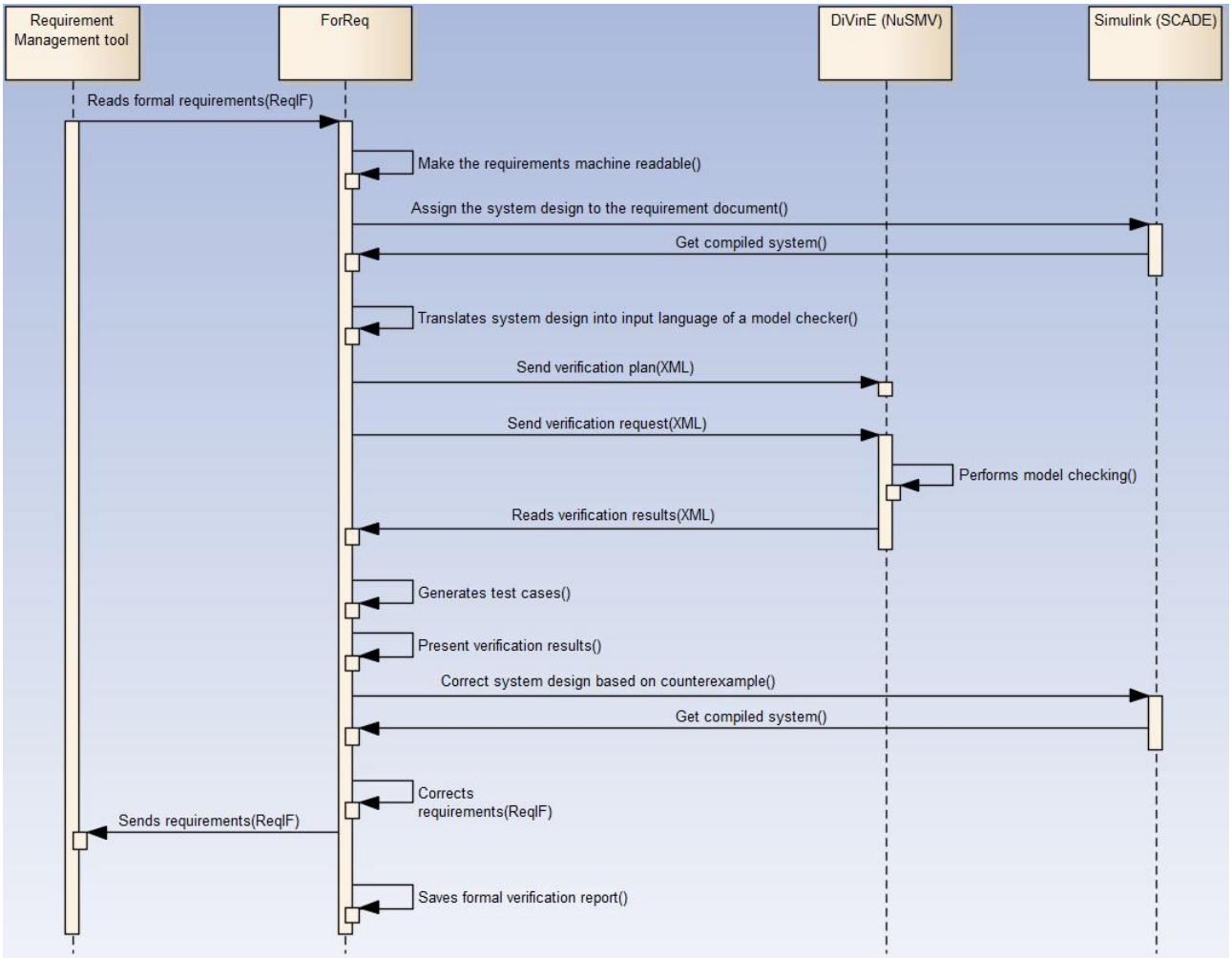

Figure 9 Requirement Verification

5 Terms, Abbreviations and Definitions

3VSE	**Three View System Engineering – Honeywell model-based development process**
AHRS	**Attitude Heading Reference Systems**
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.**
ForReq	**Formalization of Requirements – internal Honeywell tool**
FTA	**Fault Tree Analysis**
GNSS	**Global Navigation Satellite Systems**
IRS	**Inertial Reference Systems**
kM	**Knowledge Manager – a tool from REUSE Company**
LTL	**Linear Temporal Logic**
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)**

Table 5-1: Terms, Abbreviations and Definitions

6 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.
[Farfeleder, 2011]	*DODT: Increasing requirements formalism using domain ontologies for improved embedded systems development*; **Farfeleder, S.; Moser, T.; Krall, A.; Stålhane, T.; Zojer, H.; Panis, C.**; Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2011 IEEE 14th International Symposium on, April 2011, 978-1-4244-9755-3

7 Annex I: Detailed Descriptions of the Engineering Methods

The detailed version of the engineering methods, where input and output artefacts based on the Crystal template is bellow:

EngineeringMethods-Multi-ModeNavigation

8 Annex II: Technology Base Line & Progress Beyond

While the future of engineering methods as expected to be achieved within Crystal project are detailed in the chapter 4, the current practice for each engineering methods (state of the art) is as follows:

Engineering Method: Ontology Learning		
Purpose: To create domain ontology in order to ease requirement creation.		
Pre condition	Engineering Activity	Post condition
	There is no ontology involved in the development process.	
Engineering Method: Requirement Authoring		
Purpose: To capture requirements		
Pre condition	Engineering Activity	Post condition
Customer requirements are available in any format.	The manual process and expert analysis is involved but no supporting tool and no ontology.	System high-Level requirements created.
Engineering Method: Requirement Analysis		
Purpose: To increase quality of requirements		
Pre condition	Engineering Activity	Post condition
System High-Level Requirements stored in Word or DOORS.	The manual review of the requirements and expert analysis is involved but no supporting tool and no ontology.	System High-Level Requirements stored in Word or DOORS, which is approved
Engineering Method: Requirement Analysis and Verification		
Purpose: To make requirements machine readable		
Pre condition	Engineering Activity	Post condition
	There is no formal verification of the requirements involved.	
Engineering Method: Operational Analysis		
Purpose: Identify the system operations.		
Pre condition	Engineering Activity	Post condition
Customer requirements are available.	Manual process usually in Word.	System boundaries, operational scenarios and mission scenarios.
Engineering Method: Functional Analysis		
Purpose: Identify what the system must do to accomplish the work.		
Pre condition	Engineering Activity	Post condition
System High-Level Requirements stored in Word or DOORS.	In System Design.	Functional Model created.
Engineering Method: Safety Analysis		
Purpose: To ensure the safety of the system.		

Pre condition	Engineering Activity	Post condition
System design is created.	Manual process in Word.	Safety analysis is performed.
Engineering Method: System Design		
Purpose: To design the system.		
Pre condition	Engineering Activity	Post condition
System High-Level Requirements stored in Word or DOORS.	Create system design in Simulink.	Executable system design is created.
Engineering Method: Verify design against requirements		
Purpose: To verify if the system design satisfies a set of given requirements.		
Pre condition	Engineering Activity	Post condition
System High-Level Requirements stored in Word or DOORS. System Design in Simulink.	Manual review process. (Testing is performed only against low-level requirements.)	Verification approval that the requirements are satisfied in the system design.
Engineering Method: Architectural Analysis		
Purpose: Defines how the system will implement the functional model and system requirements.		
Pre condition	Engineering Activity	Post condition
System High-Level Requirements stored in Word or DOORS. System Design in Simulink.	Performed manually in Word or in system design.	Specific implementation and allocation of the software and hardware parts of system design.
Engineering Method: Code Generation		
Purpose: To generate the source code.		
Pre condition	Engineering Activity	Post condition
Executable system design is created.	Generate the source code from Simulink.	Source code is created.