

DOMAIN ENGINEERING PROCESS ASSESSMENT REPORT

GMV-CORDET-RP-002 DRAFT A
CORDET

Prepared by: Elena Alaña - GMV
Ana-Isabel Rodríguez - GMV

Approved by: Ana-Isabel Rodríguez
GMV Project Manager

Authorized by: Ana-Isabel Rodríguez
GMV Project Manager

Internal Code: GMVSA 20955/08

Version: Draft A

Date: 25/08/2008

DOCUMENT STATUS SHEET

Version	Date	Pages	Changes
Draft A	25/08/2008	35	First draft version

TABLE OF CONTENTS

1. INTRODUCTION	5
1.1. PURPOSE AND SCOPE	5
1.2. APPLICABLE DOCUMENTS	5
1.3. REFERENCE DOCUMENTS	5
2. COMPARISON OF THE TRACKS FOLLOWED BY DOMENG AND CORDET AND RETURN OF EXPERIENCE	7
2.1. TRACK FOLLOWED IN CORDET	7
2.2. TRACK FOLLOWED IN DOMENG	8
2.3. COMMONALITIES AND DIFFERENCES	9
3. COMPARISON OF THE MAIN MODELS, DESIGNS AND IMPLEMENTATIONS DIFFERENCES AND RETURN OF EXPERIENCE	10
3.1. MODELS, NOTATIONS AND TOOLS	10
3.2. COMMONALITIES AND DIFFERENCES	10
4. GUIDELINES FOR DOMAIN ANALYSIS: METHODS AND TOOLS	14
4.1. REQUIREMENTS ANALYSIS	14
4.1.1. REQUIREMENTS MODELS	14
4.2. CONTEXT ANALYSIS	17
4.2.1. CONTEXT MODEL	17
4.2.1.1. BLOCK DIAGRAM	17
4.2.1.2. INTERNAL BLOCK DIAGRAM	19
4.3. DOMAIN MODELLING	20
4.3.1. INFORMATION MODEL.....	20
4.3.1.1. BLOCK AND INTERNAL BLOCK DIAGRAMS	20
4.3.1.2. USE CASE DIAGRAM.....	21
4.3.1.3. ACTIVITY DIAGRAM.....	21
4.3.2. FEATURE MODEL.....	23
4.3.2.1. FEATURE DIAGRAMS.....	23
4.3.3. OPERATIONAL MODEL.....	24
4.3.3.1. STATE-MACHINE DIAGRAMS.....	24
4.3.3.2. SEQUENCE DIAGRAM	26
4.3.4. DICTIONARY	27
4.4. MODELS VALIDATION	28
4.4.1. VALIDATION OF REQUIREMENTS, CONTEXT, INFORMATION AND OPERATIONAL MODELS.....	28
4.4.2. VALIDATION OF FEATURE MODELS.....	28
5. GUIDELINES FOR DOMAIN DESIGN.....	31
6. CONCLUSIONS DEDUCED FROM THE TWO STUDIES ON THE DOMAIN ENGINEERING	32

LIST OF TABLES AND FIGURES

Table 1-1: Applicable Documents.....	5
Table 1-2: Reference Documents.....	6
Table 3-1: DOMENG/CORDETs Models, Notations and Tools.	13
Table 4-1: Elements of the requirements diagram included in DOMENG diagrams	17
Table 4-2: Elements of block diagrams included in DOMENG diagrams	18
Table 4-3: Elements of activity diagrams included in DOMENG diagrams	22
Table 4-4: Feature and group cardinality of XFeature diagrams.....	23
Table 4-5: Elements of state machine diagrams included in DOMENG diagrams	26
Table 4-6: Elements of sequence diagrams included in DOMENG diagrams.....	27
Figure 4-1: Requirements Diagram	15
Figure 4-2: Requirements Outline View	16
Figure 4-3: Block diagram.....	19
Figure 4-4: Context Diagram – Block diagram and internal block diagram	20
Figure 4-5: Use case diagram (Information Model)	21
Figure 4-6: Activity Diagram (Information Model)	22
Figure 4-7: Feature Model	24
Figure 4-8: State Machine Diagram (Operational Model).....	25
Figure 4-9: Sequence diagram (Operational Model)	26
Figure 4-10: jVLT – Vocabulary Learning Tool.....	27
Figure 4-11: Unsuccessful validation of Requirements, Context and Domain Models	28
Figure 4-12: Successful validation of Requirements, Context and Domain Models.....	28
Figure 4-13: Successful generation of Feature Metamodel	29
Figure 4-14: Successful validation of Feature model.....	29
Figure 4-15: Error generating the metamodel.....	29
Figure 4-16: Error Validating the model	29
Figure 5-1: Approach to define reference architectures.....	31

1. INTRODUCTION

1.1. PURPOSE AND SCOPE

This report is written as part of the ESA study on Component Oriented Development Techniques (CORDeT), AO/1-5237/06/NL/JD ([AD.1] and [AD.2]).

The document is the output of WP303 "Feedback and synthesis w.r.t. DOMENG". The report covers the next set of activities:

- Preparation of a guideline for domain engineering process, methods and tools. These guidelines will facilitate the understanding of domain engineering models and fix the basis to be able to run the domain engineering process again (next iterations) in future ESA studies
- Comparison of the tracks followed by CORDET and DOMENG studies and return of experience.
- Comparison of the main models, designs and implementations differences and return of experience.
- Conclusions deduced from the two studies on the domain engineering.

1.2. APPLICABLE DOCUMENTS

The following documents, of the exact issue shown, form part of this document to the extent specified herein. Applicable documents are those referenced in the Contract or approved by the Approval Authority. They are referenced in this document in the form [AD.X]:

Reference	Title	Code / Version
[AD.1]	Component Oriented Development Techniques: Statement of work	EME-012 – issue 1.1 – 28 July 2006
[AD.2]	Invitation to tender number AO/1-5237/06/NL/JB – Component Oriented Development Techniques	ESA /IPC (2001) 11 – Item no. 01.1EM.06
[AD.3]	Component Oriented Development Techniques. Invitation To Tender AO/1-5237/06/NL/JB: Part 1 Technical Proposal Part 2 Management Proposal Part 3 Financial and Contractual Proposal	ASP-06-EL/PE/S-157, Issue 01, October 16 th , 2006.
[AD.4]	ISO AMD 1 and AMD 2 2004, Corrigenda –Information of ISO/IEC 12207:1995 Software Life cycle Processes. ISO/IEC 12207 – limited to the domain engineering process	ISO/IEC 12207 :1995

Table 1-1: Applicable Documents

1.3. REFERENCE DOCUMENTS

The following documents, although not part of this document, amplify or clarify its contents. Reference documents are those not applicable and referenced within this document. They are referenced in this document in the form [RD.X]:

Reference	Title	Code
[Ref-CORDET-1]	CORDeT – Cannes study “CORDeT Domain Acquisition report”	ASP-07-EL/PE/S-104 Issue 2.0
[Ref-CORDET-2]	CORDeT – Toulouse study “CORDeT Domain Analysis report”	CORDeT/R03 0.3 12/12/07
[Ref-CORDET-3]	CORDeT – Toulouse study “ <i>Domain Engineering Methodologies, Languages and Tools</i> ” John Favaro and Silvia Mazzini (Intecs)	CORDeT/R02 28/05/07
[Ref-CORDET-4]	CORDeT - “Domain Engineering Methodologies Survey”.	Issue 2 31/10/2007
[Ref-CORDET-5]	CORDeT – Cannes study “Domain Analysis report Whole domain level”. TAS-F	Issue 2.0 10/12/2007
[Ref-DASIA-INTECS]	“A Methodology for Space Domain Engineering”. Silvia Mazzini. Intecs	28/05/2008
[Ref-UML]	Unified Modelling Language (UML) Website: http://www.omg.org/uml and http://www.uml.org	-
[Ref-UML2]	UML2 Website: http://sparxsystems.com.au/resources/uml2_tutorial/index.html	-
[Ref-SYSML]	OMG SysML Specification v. 1.0 (Final Adopted Specification) Website: http://sysml.org/docs/specs/OMGSysML-FAS-06-05-04.pdf	May 2006
[Ref-TOPCASED]	TOPCASED tool Website: http://www.topcased.org	-
[Ref-XFEATURE]	XFeature Website: http://www.pnp-software.com/XFeature	-
[DAR_MoM]	Minutes of Meeting – Domain Analysis Review (CORDET and DOMENG)	17-18/Dec/2007

Table 1-2: Reference Documents

2. COMPARISON OF THE TRACKS FOLLOWED BY DOMENG AND CORDET AND RETURN OF EXPERIENCE

Up to now, several studies have investigated about component-based software, reusable assets and the development of generic architectures. In this context, DOMENG (Framework for DOMain ENGINEering) and CORDeT (Component Oriented Development Techniques) represent two parallel projects that apply the Domain Engineering Process to the space domain.

The first step followed in both projects deals with the definition of a process methodology. During the definition, a set of diagrams, notations and tools are selected to represent the different models along the whole Domain Engineering Process: domain analysis, domain design and domain implementation activities, [Ref-CORDET-4]. The selections made in CORDeT and DOMENG are collected in the Table 3-1 of section 3.2. Besides, Chapter 3 describes the main commonalities and differences of the approaches followed.

Once the models and tools have been chosen, the domain under studied has to be defined. In this case both projects have differed:

- DOMENG has focused on the whole space domain (at very high level), and also in a concrete subsystem: the HMS ("Health Management System"). HMS has been analyzed deeper.
- On the one hand, CORDeT has been performed a detail domain analysis applied on a reduced part of the OBSW domain (DH framework and the AOCS framework). Another domain analysis (not so detailed) has been done in parallel but for the global space domain (the whole software level).

Finally, the domain analysis, design and implementation activities have to be carried out. The specific tracks followed in both projects are presented in the next sections ([2.1] and [2.2]).

2.1. TRACK FOLLOWED IN CORDET

Before selecting those methods and tools to be applied in CORDeT methodology, a domain survey was carried out. This activity constitutes the first step of the Domain Engineering Process. The results provide information to define the bounds of the domain to be analysed. The conclusion extracted is oriented to focus on a general OBSW for the domain analysis, whereas the domain design and domain implementation is limited to two components of the OBSW: the TM/TC management and the AOCS applications.

Therefore, the domain boundary analysis is separated in two areas: Outside domain context and inside domain context.

The global domain analysis provides a perimeter definition, requirements and uses cases for each "macro-building blocks" to be inserted into the reference architecture. The detailed domain analysis defines how to build it internally with the objective to maximize the reuse potential for each block.

Finally, the OBSW first iteration design is defined. It is said that the generic architecture shall minimize the impacts of the four level of variability: processor module, avionics level, operational level (command and monitor) and mission level (variations mission dependant). This variability provides a first layered architecture where the layers order depends on the way that the majority of the calls are executed.

A layered architecture is outlined following a component-based architecture approach. The components are an operating system, a middleware (assigned to processor and avionics dependant layers) and several components (assigned to mission dependant layer) on top of it.

The operational concept layer is divided in two types of services: horizontal services (middleware) and definition of components (implemented as components).

2.2. TRACK FOLLOWED IN DOMENG

The DOMENG project uses the Domain Engineering Process for the space domain to establish the framework for the space systems domain engineering.

DOMENG starting point is being the ISO 12207, the ECSS Standards and the IEEE Std. 1517. Following this approach, DOMENG begins defining the methodology, models, and tools to be used.

Subsequently, the knowledge relevant to space domain is gathered and taken as input for the next activities. Information gathering was carried out through questionnaires grouped into different fields (organizational, SW product, SW development process, business, people/organizational related aspects and system engineers) and oriented to different audience (business unit people, system engineers with relevant experiences in past or on-going OBSW projects, advanced studies engineers, people studying future trends in terms of functionalities, technologies, architecture, and software engineers).

For the domain analysis activity the global space system and the HMS has been studied. HMS ("Health Management System") has been chosen to detail the Domain Engineering Process. HMS involves monitoring the operational quality of the different components and continuing operation as long as defined limits are not exceeding, but also Autonomous Recovery Actions and Prognosis.

Comparing the space domain models to the HMS ones, the HMS also includes:

- More detail context models.
- Information model: activity diagrams
- Operation model: sequence diagrams

When a specific subsystem of the OBSW domain is analyzed, the methodology has been carried out completely (with all possible models) and in an exhaustive way. However, when the entire space domain is considered it cannot be studied so deeply. This is due to the fact that analyzing the whole domain includes many subsystems with complex functionality and the analysis involves a very long process that it cannot be covered in the period of time of these projects. Hence, the decision taken for the domain analysis considers the study of a specific subsystem completely to demonstrate the propose methodology, and a first high-level iteration of the global space domain.

In addition, it has been extracted the main user needs to define an OBSW reference architecture. These user needs has been organize hierarchically and modelled through a SysML requirements diagram.

Finally, a high-level reference architecture is proposed for the whole space domain. This architecture combines a component-oriented approach with schedulability and dependability issues. The architecture can be divided into functional and software parts. The Dependability properties will be developed during the domain design phase.

As well, a preliminary Software Criticality Analysis has been done. It serves as an initial criticality review of the DOMENG Domain Analysis resulting in requirements to be added to the domain model containing the space domain boundaries, developers' needs, domain models and vocabulary. It presents the results of the performance of an SFMECA analysis chosen as the criticality analysis technique to be used to define the critical functions and the mechanisms for the prevention and handling of the identified failure modes with special attention to those critical ones. It defines the criticality categories used (presented too to be the ones to be the reference ones in any space domain) as well as the classified applicable system space domain feared events.

The output report presents the criticality categories used together with the already classified feared events that can be both used to perform the SFMECA analysis to the identified critical items and the failures effects summary.

Being a top level of SFMECA of a rather abstract (up to now) product, the attainable level of detail is not great, but it must nevertheless produce valuable inputs for the clarification of "architectural drivers" for the Space Domain Analysis and Generic Architectures definition.

2.3. COMMONALITIES AND DIFFERENCES

Here are some commonalities and differences of the tracks followed by both projects:

- Methodology.
 - Almost similar selection of models and tools, [Chapter 3.].
 - Both include information gathering activity.
 - DOMENG extracts the main user needs to define the OBSW reference architecture.
 - DOMENG includes a Preliminary Software Criticality Analysis.
- Domain Analysis:
 - DOMENG and COrDeT include a global space domain analysis.
 - DOMENG focuses on HMS whereas COrDeT on the software frameworks for the Data Handling subsystem and for the Attitude and Orbit Control subsystem.
- High-Level architecture:
 - Both DOMENG and COrDeT high-level architectures are component-oriented. DOMENG high level architecture combines a component-oriented approach with schedulability and dependability issues.
 - DOMENG high level architecture defines different levels of schedulability
 - The DOMENG concept of SW Assets Library is similar to COrDeT Framework (FW) approach, but real-time aspects in COrDeT are relegated to ASSERT RCM-based virtual machine and in DOMENG to the Virtual Bus and the Middleware Interface component.
 - In ASSERT the FW functional components are embedded with non-functional containers that endow them with real-time properties. This can be compatible for DOMENG high level architecture low-level scheduling.
 - In DOMENG high level architecture, SW Assets services are separated into three layers: Mission Layer (ML) Services, Operational Layer (OL) Services and Avionics Layer (AL) Services. COrDeT DH and AOCS Frameworks should map into the Operational Layer (OL) Concept.
 - The functional components of the OBSW are mapped to High-level SUs of different types: ML High-level SUs, OL High-level SUs and AL High-level SUs.
- Others:
 - In DOMENG we have completed the domain analysis models (context, features, information and operational models), which have allowed us to provide a first approach to a generic architecture.

3. COMPARISON OF THE MAIN MODELS, DESIGNS AND IMPLEMENTATIONS DIFFERENCES AND RETURN OF EXPERIENCE

3.1. MODELS, NOTATIONS AND TOOLS

SysML is used as the modelling language for the domain analysis. Topcased tool was selected to develop SysML diagrams, and XFeature is selected to design feature diagrams.

The models included in both projects are the following:

- The COrDeT domain analysis output provides a requirements model, a context model, a feature model, a description of all the variation points in the domain and a dictionary.

A considerable difference in relation to DOMENG is the use of Mindmap tool. It is used to perform a hierarchical decomposition of the domain. A mind-map is an image-centred diagram that represents semantic or other connections between portions of information having a brainstorming approach. The mind-map meta-model is transformed into a SysML model. The SysML entities are mapped to the corresponding Mindmap artifacts.

- DOMENG also includes SysML information models and operational models.

A complete specification of the models, notations and tools used are collected in Table 3-1.

Note: Diagrams can be designed in more detail when a concrete subsystem is analyzed. In these cases, internal block diagrams, activity diagrams and sequence diagrams are exhaustive. In addition, it is easier to analyze the context and behaviour than when the complete space domain is considered.

Chapter 4 compiles some guidelines for the diagrams designed. It explains the objective of the diagrams, the meaning of their elements and some problems found.

3.2. COMMONALITIES AND DIFFERENCES

The most remarkable aspects of the domain analysis in COrDeT with respect to DOMENG are the following:

- Some COrDeT finding are injected into DOMENG:
 - The analysis of the variation points of COrDeT has allowed DOMENG to add new features to the DOMENG Feature Model.
 - Some COrDeT context diagrams are reflected in new relationships in the internal blocks diagrams of the DOMENG Context Model.
 - Some definitions listed in COrDeT dictionary have been incorporated to DOMENG dictionary.
- Tools:
 - DOMENG and COrDeT uses Topcased tool for SysML diagrams.
 - DOMENG and COrDeT uses XFeature tool for feature diagrams.
 - COrDeT also uses Mindmap tool for the decomposition of the domain.
- The proposed domain methodology has not been followed in COrDeT:

- Difficulties in modelling the requirements from the knowledge acquired during the domain knowledge acquisition phase.
- Some problems found at using SysML and Topcased.
- The information and operational models are not provided.
- No standard notation applied to the OBSW first iteration design.
- Both projects have found some problems using Topcased tool.
 - COrDeT: Difficult to “iterate” on the model [Ref-CORDET-5]. It is very difficult to iterate on the model using the Topcased tool, it is very difficult to reorganize the model when something appears to be not correct, for example moving a set of related use-case between packages is a quite complex operation. Topcased tool does not support correctly according to us an iterative modeling which is mandatory for the kind of activities we perform in the study.
 - COrDeT: Size of diagrams explodes [Ref-CORDET-5]. Most of the diagrams require to extend the page size up to A0 which does not permit a correct human exploitation of the diagrams.
 - CORDET: Hierarchical organization [Ref-CORDET-5]. There is no information on the best way to organize models both in the SysML or UML specifications, and Topcased documentation. This question is very important in order to have an efficient model for both human manipulation and exploitation and automated transformations. Organization of the model gathers the possible split into separate model files, the package strategy in the model, the diagram location, number,...
 - Delete elements from models takes a long time, it is not instantaneous.
 - Some information is not printed in the diagrams. It can only be accessed through the properties window.

DOMENG - GSTP				CORDeT		Compatibility
Phase	Model	Notation	Tools	Track 1 - P&P	Track 2 - Thales	
				Notation & tools	Notation & tools	
Domain Analysis	Requirements Model	SysML Requirement diagrams.	Topcased	<p>Domain Analysis shall be analogous to a conventional requirements definition phase where natural language is used. As far as possible SysML models shall be used to represent these requirements.</p> <p>Features represented in a feature model.</p> <p>XFeature tool in "FD Configuration" is chosen to build the feature model. "FD Configuration" is defined in Assert project.</p> <p>Topcased tool to build SysML diagrams.</p> <p>Construction of a domain dictionary used to define terms that can be used to formulate shared properties.</p>	<p>Domain Analysis shall be analogous to a conventional requirements definition phase where natural language is used. As far as possible SysML models shall be used to represent these requirements.</p> <p>Features represented in a feature model.</p> <p>XFeature tool in "FD Configuration" is chosen to build the feature model. "FD Configuration" is defined in Assert project.</p> <p>Topcased tool to build SysML diagrams.</p> <p>Construction of a domain dictionary used to define terms that can be used to formulate shared properties..</p>	<p>DOMENG and CORDET Cannes are based in FODA approach.</p> <p>Features are represented through a feature model using XFeature tool.</p> <p>DOMENG will use a set of SysML models in order to define requirements and the bounds of the domain. All these models are developed using Topcased. P&P and Thales main goal is to express all the requirements in natural language. Later, as far as possible they will be represented using SysML (Topcased). So, all the approaches are compatible.</p>
		SysML Use-Case diagrams.	Topcased			
	Context Model	Context and Data-flow diagrams. SysML Block (Block definition and internal block) diagrams.	Topcased			
	Feature Model	UML static structure diagram or SysML block diagrams.	XFeature (Eclipse plug-in)			
	Dictionary	-	Word			
	Information Model	SysML Use-Case and Block diagrams	Topcased			
	Operational Model	Interaction and State diagrams. SysML Parametric diagrams	Topcased			

DOMENG - GSTP				CORDeT		Compatibility	
Phase	Model	Notation	Tools	Track 1 - P&P		Track 2 - Thales	
				Notation & tools		Notation & tools	
Domain Design	Domain Architecture	UML2 <u>Functional view</u> : Class structure and state chart diagrams. <u>Non-Functional view</u> : Timing, interaction overview and activity diagrams.	Topcased/Eclipse(Tog ether plug-in)	UML2 models based on FW Profile for CORDET Software Framework. UML2 meta-model based on the RCM for CORDET System Family.	Topcased customized with the FW Profile Eclipse Plug-in. Topcased customized with RCM metadata model.	-	GMV, Thales and P&P consider UML2 to design the architecture and Topcased is chosen as design tool.
	Transformation Rules	-	Together (Eclipse)	-		-	
	Domain Documentation and Architecture Evaluation	-	Word	-		-	
	Platform Architecture	UML2	Topcased/Eclipse(Tog ether plug-in)	-		-	

Table 3-1: DOMENG/CORDETs Models, Notations and Tools.

4. GUIDELINES FOR DOMAIN ANALYSIS: METHODS AND TOOLS

Domain Engineering should be a continuous and iterative process, and the knowledge concerning the domain should be maintained and updated according to new experiences, trends and feedback from application engineering process. Therefore, it would be useful to compile the methods and tools used during this process. This way, DOMENG and CORDeT results may be reused for either understand the models or to continue developing the Domain Engineering through another iteration.

Domain Analysis is the activity that discovers and formally describes the commonalities and variabilities within a space domain. It is divided into three different phases to get the final domain model:

- Requirements analysis [4.1].
- Context analysis [4.2].
- Domain modelling [4.3].

Next subsections collect some guidelines for the different diagrams generated for this activity.

4.1. REQUIREMENTS ANALYSIS

The objective of the requirements analysis is to model requirements within a logical organization.

- Inputs: List of domain requirements and needs.
- Outputs: Requirements model.
- Notation: SysML [Ref-SYSML].
- Tools: Topcased [Ref-TOPCASED].

4.1.1. REQUIREMENTS MODELS

Requirements models are generated using Topcased Tool. The hierarchy describes requirements contained in a specification.

Figure 4-1 shows an example of a requirements diagram (including derive and containment relations). This kind of diagram provides a clear view of the relations and the organization among requirements. The content of all the information is directly visible from the printed diagrams.

When a change is done in any requirement field, it is automatically changed in all the diagrams where this requirement appears. Therefore, each diagram is not generated in isolation. Each of them is interconnected with those with which it has dependencies. A trace is kept among different diagrams.

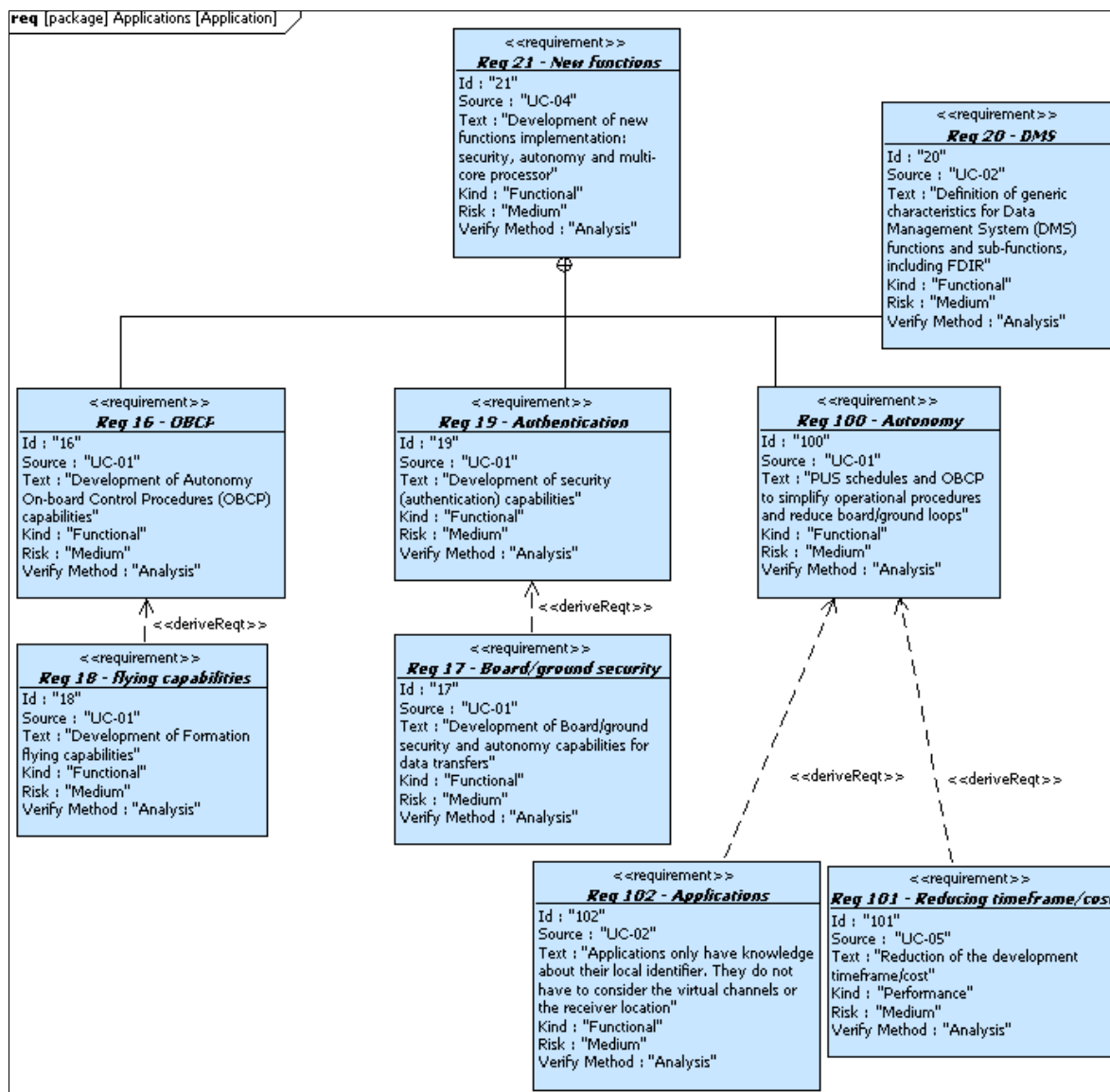


Figure 4-1: Requirements Diagram

The organization of requirements is also shown in the Topcased outline view. This view let the user check the different relations among requirements [Figure 4-2].

If a requirement is selected with double-click in the outline view, then it allows to access to all the diagrams where the block is present. If the requirement exists only in a single diagram, it is open automatically. When more than one exists, the selection window appears, and one of them has to be selected. When the element does not exist in any diagram, nothing happens. This behavior occurs not only in requirement diagrams but also in all the different diagrams that can be generated using Topcased.

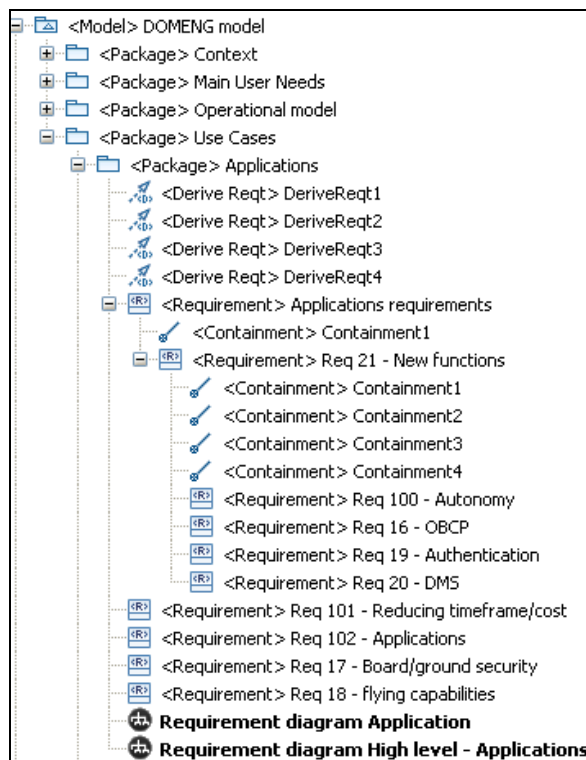
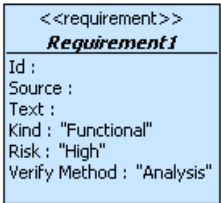









Figure 4-2: Requirements Outline View

These diagrams are mainly composed by requirements fields and the relations among them. Table 4-1 summarizes the main relations and the different fields to define a requirement [Ref-SYSML]:

Element	Icon	Fields	Description
Requirements Field		Name	Short name for the requirement
		ID	The requirement identification
		Source	Traceability (E.g. In DOMENG, traceability to use cases)
		Text	Full description of the requirement
		Priority	High, Medium or Low
		Validation Method	Analysis, Inspection, Demonstration or Test
Relations between elements	 Containment  Satisfy  DeriveReq  Verify  Refine  Copy  Trace	Containment	Dependency relationship where a client requirement contains the supplier requirement
		Satisfy	Dependency relationship between a requirement and a model (design or implementation) that fulfils the requirement
		DeriveReq	Dependency relationship between two requirements in which a client requirement can be derived from the supplier requirement

Element	Icon	Fields	Description
		Verify	Relationship between a requirement and a test case that can determine whether a system fulfil the requirement
		Refine	Used to describe how a model element or a set of model elements can be used to further refine a requirement
		Trace	General purpose relationship between a requirement and any other model element

Table 4-1: Elements of the requirements diagram included in DOMENG diagrams

DOMENG traces textual requirements to diagrams manually and stored the results in a table. Traceability between requirements and the design is also stored in a table. This table shows if the requirements are total, partial or not addressed.

4.2. CONTEXT ANALYSIS

Context analysis tries to establish the bounds of the domain, the relationships with other domain (inputs and outputs) and the scope of the analysis.

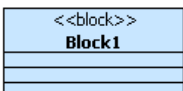
- Inputs: Standards, application framework (information gathering in order to identify features).
- Outputs: A context or data-flow diagram where the domain is placed relative to higher, lower, and peer level domain. Variabilities are identified using several diagrams. In each one of the possible diagrams, the context is identified. This diagram also shows the data-flows between the target domain and other abstractions.
- Notations: SysML Block diagrams [Ref-SYSML]: Block definition diagrams and Internal block diagrams.
- Tool: Topcased [Ref-TOPCASED].

4.2.1. CONTEXT MODEL

Block and internal block diagrams are used to represent the context and structure of the system. But also, block diagram provides a view of the different components of a system and the connections to other domains. Internal block diagrams are more specific and usually used to specify the internal structure of a system or the relations among properties of a block.

4.2.1.1. Block diagram

Table 4-2 collects some elements contained in a block definition diagram (the ones used to define the context models). A complete description is available in [Ref-SYSML].

Element	Icon	Description
Block		Collection of features (structural and behavioural) to describe the structure of a system or element. Blocks are the basic structural elements. Blocks may include a structure of connectors between its properties to show the relations among them.











Element	Icon	Description	
Actor	 Actor1	Actors specify roles played by users on any other system that interacts with the subject	
Port		Ports specify services that the owning block offers to its environment as well as the services that the owning block requires of its environment	
Flow Port		It specifies the input and output items that may flow between a block and its environment	
Relationships between elements	 Association  Generalization  InterfaceRealization  Usage  Dependency  Connector  Containment	Association	Semantic relationship between typed instances
		Generalization	It relates a specific classifier to a more general classifier, and is owned by the specific classifier
		InterfaceRealization	Relationship between a classifier and an interface that implies that the realizing classifier conforms to the contract specified by the interface
		Usage	Relationship in which one element requires another for its full implementation or operation
		Dependency	Relationship between one or more model elements that requires other model elements for their specification or implementation
		Containment	Dependency relationship where a client block contains the supplier block

Table 4-2: Elements of block diagrams included in DOMENG diagrams

Below is an example of a block diagram. Apart from the previous elements comments can be added (block in orange).

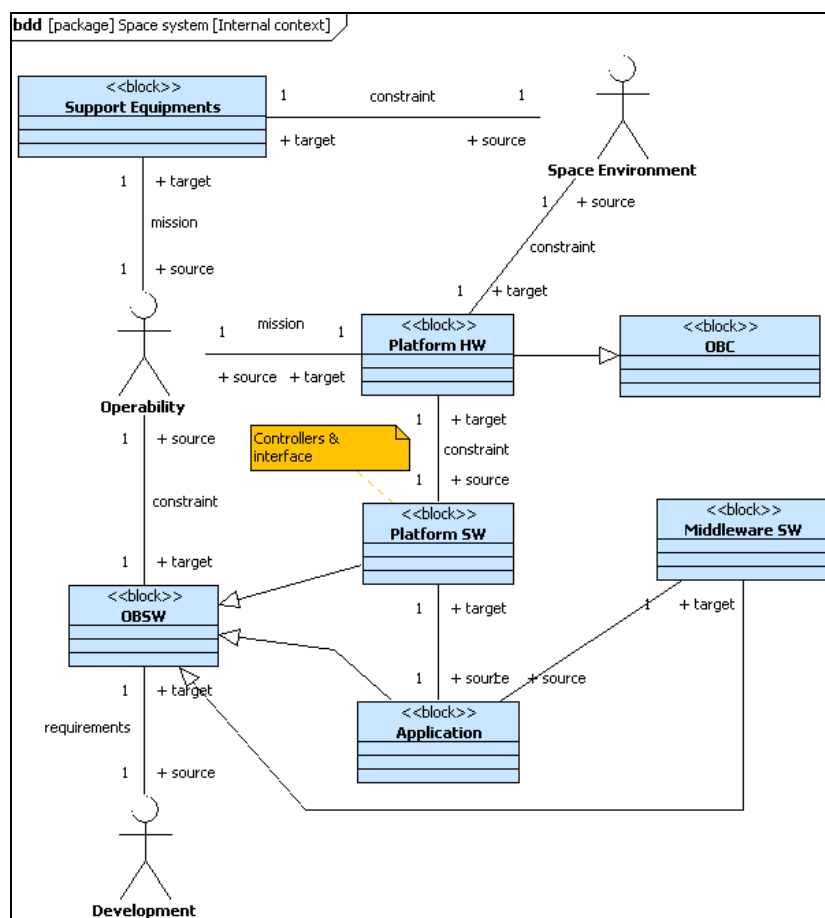


Figure 4-3: Block diagram

Figure 4-3 shows the relations among components of the system but also how the external entities interact with them.

4.2.1.2. Internal block diagram

Each block can be detailed in an internal block diagram. Internal blocks describe the internal structure of a block in terms of its properties and connectors. As an example, Figure 4-4 collects on the left a block diagram with a set of block properties, on the right it is represented the corresponding internal block diagram. In this last diagram the relations among the different properties are shown.

The block field has an arrow on the upper-right corner. When this arrow is clicked its corresponding internal block diagram is opened.

In the same way than for requirements diagrams, when a block is selected in the outline view and it has not only a block but also an internal block diagram, then the selection window appears to let the user select the concrete diagram he wants to open.

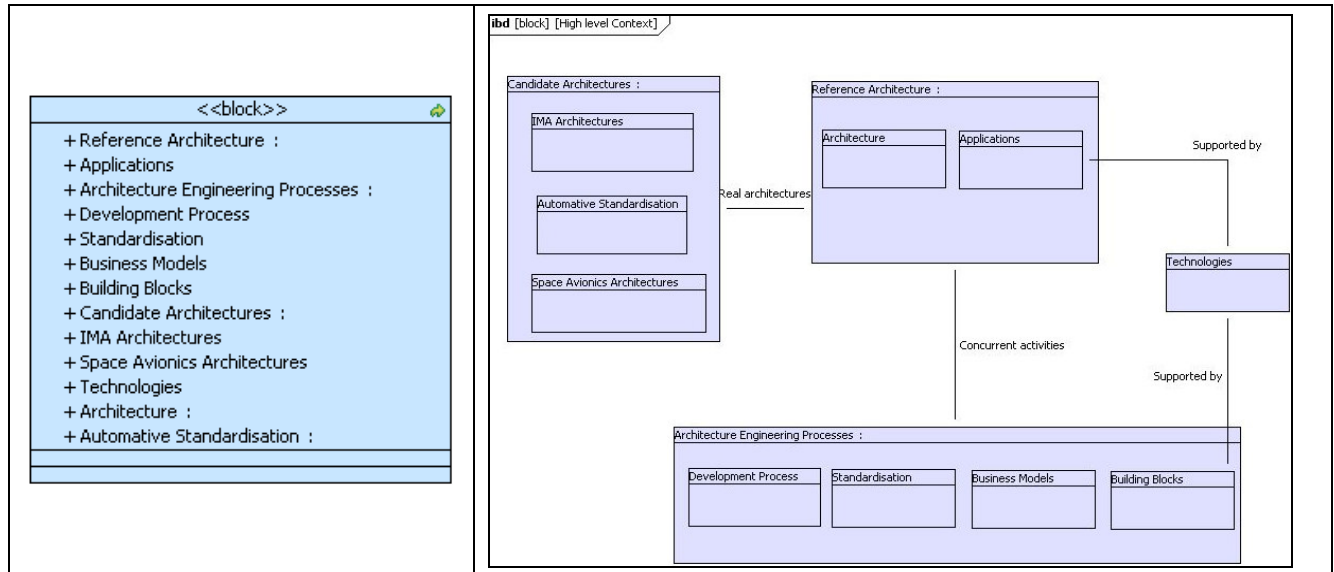


Figure 4-4: Context Diagram – Block diagram and internal block diagram

The use of block diagrams or internal diagrams depends on the information to show. Internal blocks represent the relationship between properties of a given block. The same information could not be represented in both types of diagrams.

4.3. DOMAIN MODELLING

Domain Modelling is integrated by the information model, the feature model, the operational model and the dictionary.

4.3.1. INFORMATION MODEL

This model consists of the domain's entities or abstractions, and the relations between them.

- Inputs: Features and context model.
- Outputs: A more refinement block diagram than the one designed during the context analysis phase. Use-case diagrams to specify the information exchanged among end-users and the abstracted application.
- Notation: SysML use case, block and internal block diagrams [Ref-SYSML].
- Tools: Topcased [Ref-TOPCASED].

4.3.1.1. Block and Internal Block Diagrams

More specific block [4.2.1.1] and internal block diagrams [4.2.1.2] must be defined in order to specify the context to be modelled.

The most used diagrams are the internal ones to show data flow in more detail. In DOMENG internal block diagrams include the software, hardware interface and hardware.

4.3.1.2. Use case Diagram

Use cases are defined taking into account the requirements and the context diagrams defined.

In addition, functional requirements can be modelled using use cases. Non-functional requirements cannot be represented in a use case but they can be modelled using other model diagrams.

This kind of diagram compiles the main use cases and the interaction of external actors with them. Actors invoke or obtain data from the system.

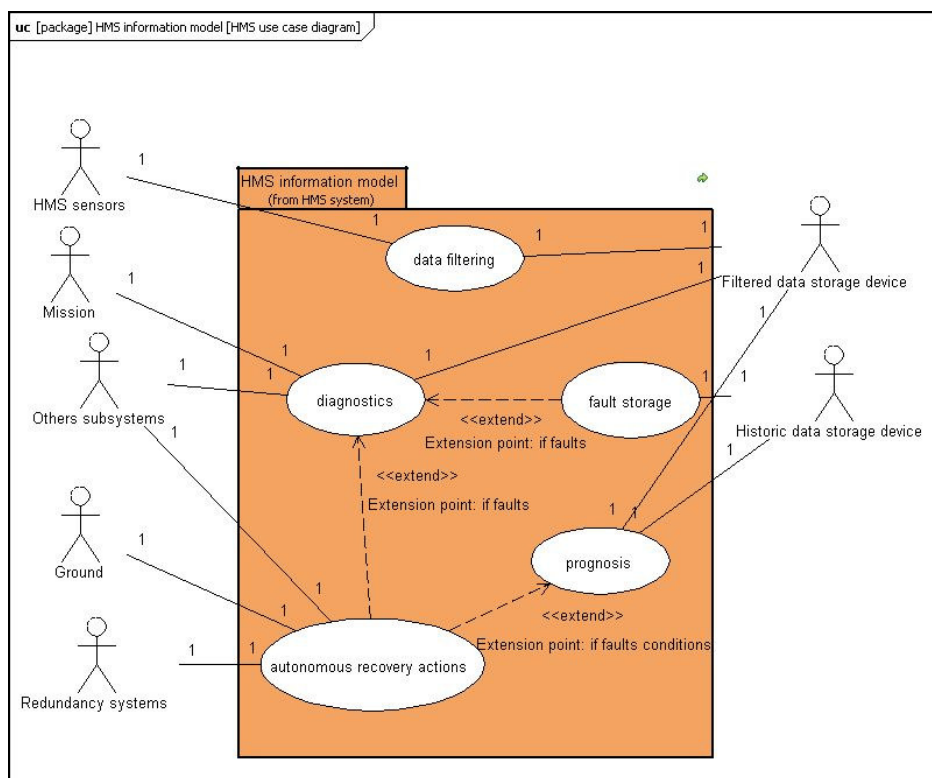


Figure 4-5: Use case diagram (Information Model)

A drawback of designing use case diagrams with Topcased is that the name of the association relations is not directly visible in the printed diagrams. This information is accessible when the link is selected. Then, the name of the association appears in the properties window.

4.3.1.3. Activity Diagram

For each use case its corresponding activity diagram can be defined. They facilitate the understanding of the data flow and how an actor is involve in the use case.

Figure 4-6 represents the activity diagram which corresponds to the "prognosis" use case represented in the previous figure [Figure 4-5].

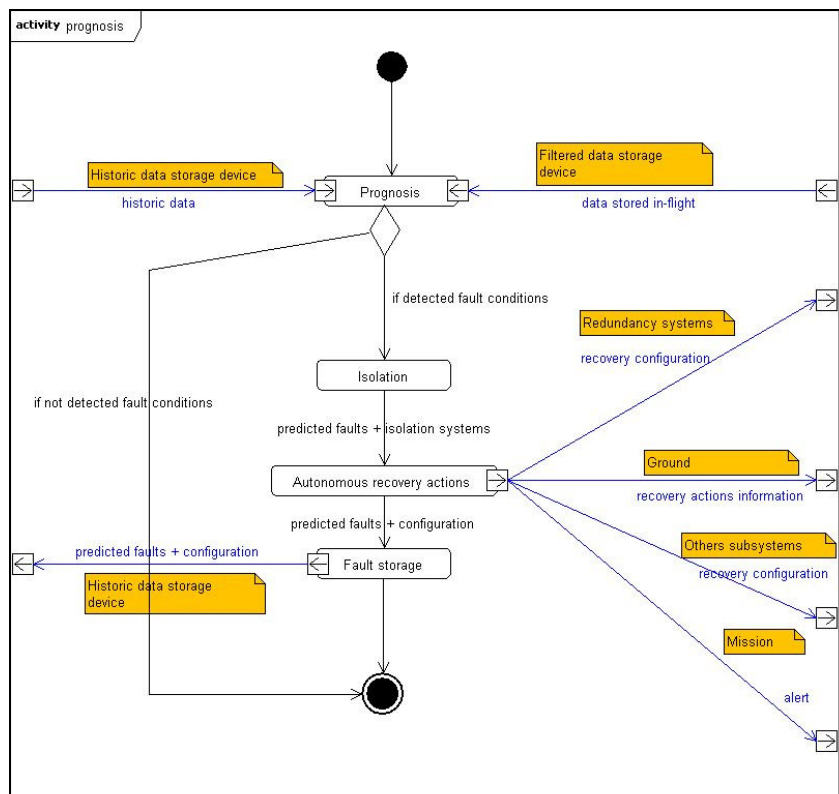


Figure 4-6: Activity Diagram (Information Model)

The main elements used in DOMENG are [Ref-SYSML]:





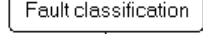

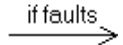

Element	Icon	Description
Initial Node		Control node where flow starts when the activity is invoked Units of flow are called tokens
Final Node		Final node terminates a flow
Decision Node		Control node that chooses between outgoing flows. Decision is based on guards. Guards expressions can be applied on all flows.
Join/Fork		Join nodes synchronizes multiple flows Fork nodes splits a flow into multiple concurrent flows
Call Operation Action		Action that transmits an operation call request to the target object
Input and output pins		Input pin holds input values to be consumed by an action Output pin holds output values produced by an action
Control Flow		Edge that starts an activity node after the previous one is finished
Object Flow		Activity edge that can have objects or data passing along it

Table 4-3: Elements of activity diagrams included in DOMENG diagrams

4.3.2. FEATURE MODEL

The feature model captures these common features and differences of the applications in a domain and their relationships.

- Inputs: Knowledge gathered among experts, end-users, etc.
- Outputs: Feature diagram, feature definition, composition rules, and rationale for rules.
- Notation: There is no specific standard to represent feature models. The most used (FODA notation) is the following. Features are represented in a tree-like structure. Each node represents a feature and its children nodes are the sub-features. There exist three different types of features:
 - Mandatory.
 - Alternative.
 - Optional.

A possibility is to represent the hierarchical model as a UML static structure diagram or SysML block diagram.

- Tools: XFeature [Ref-XFEATURE].

4.3.2.1. Feature Diagrams

Feature diagrams use "FD configuration" of XFeature tool. The diagrams have a tree structure and their main characteristics are:

- Node: represents a feature.
- Children-nodes: sub-features of the parent-node.

As well, cardinality is a basic element of XFeature diagrams:

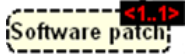

Element	Icon	Description
Feature Cardinality <a..b>		(Black boxes) Minimum and maximum number of instances of that feature in the application. Feature cardinality can be assigned to feature nodes. When no feature cardinality is assigned, <1..1> is the default value.
Group Cardinality <a..b>		Minimum and maximum number of sub-features that the system must include.

Table 4-4: Feature and group cardinality of XFeature diagrams

DOMENG Features can be mandatory or optional:

- Mandatory: These nodes are mapped to solid-line boxes.
- Optional: Nodes mapped to dashed dashed-line boxes. They are part of a group node.

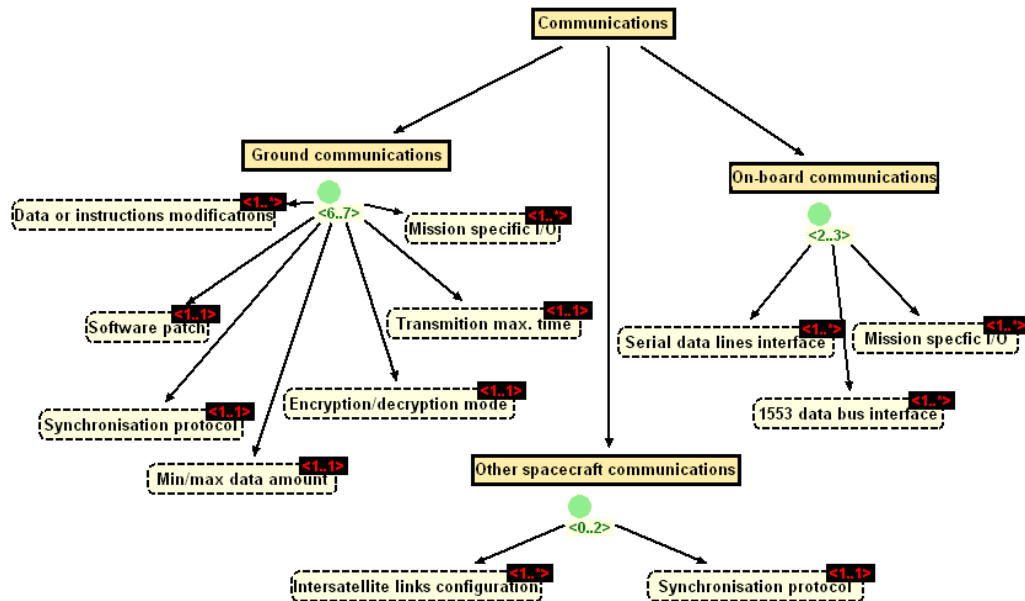


Figure 4-7: Feature Model

Figure 4-7 shows part of a feature diagram. This is an example of the meaning or identification of the previous concepts:

- "Communication" is a mandatory node which has three mandatory children-nodes (Ground communications, on-board communications and other spacecraft communications). In this cases no feature cardinality is set, so one instance (<1..1>) is the default value.
- "Other spacecraft communications" has a group node with two children nodes. The group cardinality is <0..2> one, two or even none sub-features may be defined. Hence, both sub-features are optional. Indeed, for Intersatellite links configuration could be just one instance or more than one. However, for synchronization protocol just one instance.

4.3.3. OPERATIONAL MODEL

The operational model describes the control and data-flow in the application domain, the relationships among objects in the information model and the feature model.

- Input: Context model, information model, feature model, technology of the domain.
- Outputs: interaction diagram, state diagrams.
- Notation: SysML Block, parametric, state machine and sequence diagrams [Ref-SYSML].
- Tools: Topcased [Ref-TOPCASED].

4.3.3.1. State-Machine Diagrams

State-machine diagrams support event-based behaviour of part of a system. System states are interconnected through one or more transition arcs. So that, the diagram shows the sequence of states followed in respond to certain events. Different event types are possible: change events, time events and signal events.

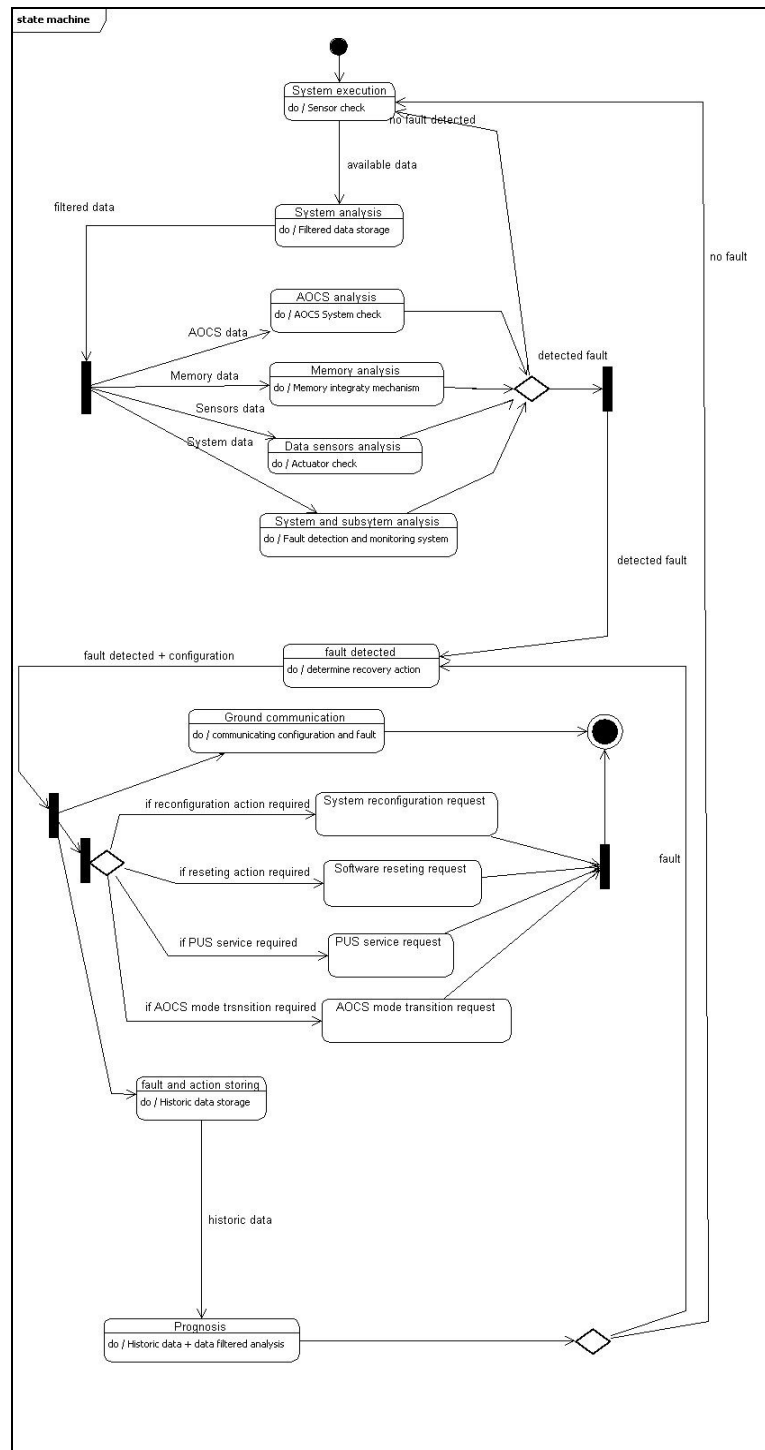


Figure 4-8: State Machine Diagram (Operational Model)

The main elements contained in DOMENG state machine diagrams are collected in Table 4-5 [Ref-SYSML]:



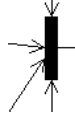

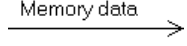
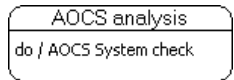
Element	Icon	Description
Initial		Default vertex that is the source for a single transition to the default state of a composite state
Final State		Vertex to terminate a flow
Join/Fork		Join vertices serve to merge several transition emanating from source vertices in different orthogonal regions. Fork vertices serve to split an incoming transition into two or more transitions terminating on orthogonal target vertices.
Choice		Choice vertices result in the dynamic evaluation of the guards of the triggers of its outgoing transitions
Transition		Relationship between a source vertex (state) and a target vertex
State		Situation during which some invariant condition holds. Kinds of states: simple, composite and submachine.

Table 4-5: Elements of state machine diagrams included in DOMENG diagrams

4.3.3.2. Sequence Diagram

These diagrams provide representations of the sequence of messages that are exchanged based on behavior: They represent flow/control and describe interactions between system components.

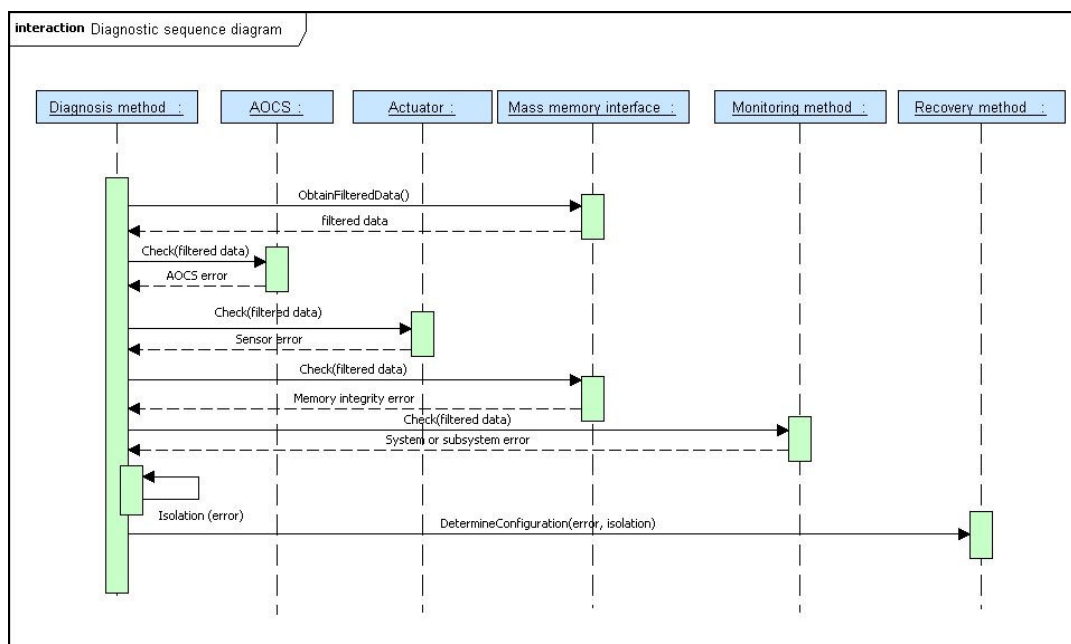


Figure 4-9: Sequence diagram (Operational Model)

The main elements contained in DOMENG sequence diagrams are [Ref-SYSML]:

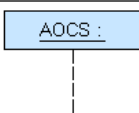
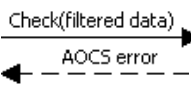

Element	Icon	Description
Lifeline		Roles or object instances that participate in the sequence being modelled
Call (Synch or Asynch) and Reply Message		Calls represent an operation/method that the receiving object's class implements and it is associated with a CallEvent
Execution		State of the system or model when it is running

Table 4-6: Elements of sequence diagrams included in DOMENG diagrams

4.3.4. DICTIONARY

Domain dictionary collects the Domain Engineering Process terminology.

CORDeT stores the vocabulary in plain text. However, the use of some tool facilitates the user actions: search, delete, save, etc.

The DOMENG vocabulary has been collected using jVLT-Vocabulary Learning Tool (Version 1.0.2). The vocabulary can be exported to an excel file.

The user interface of this tool is shown in Figure 4-10:

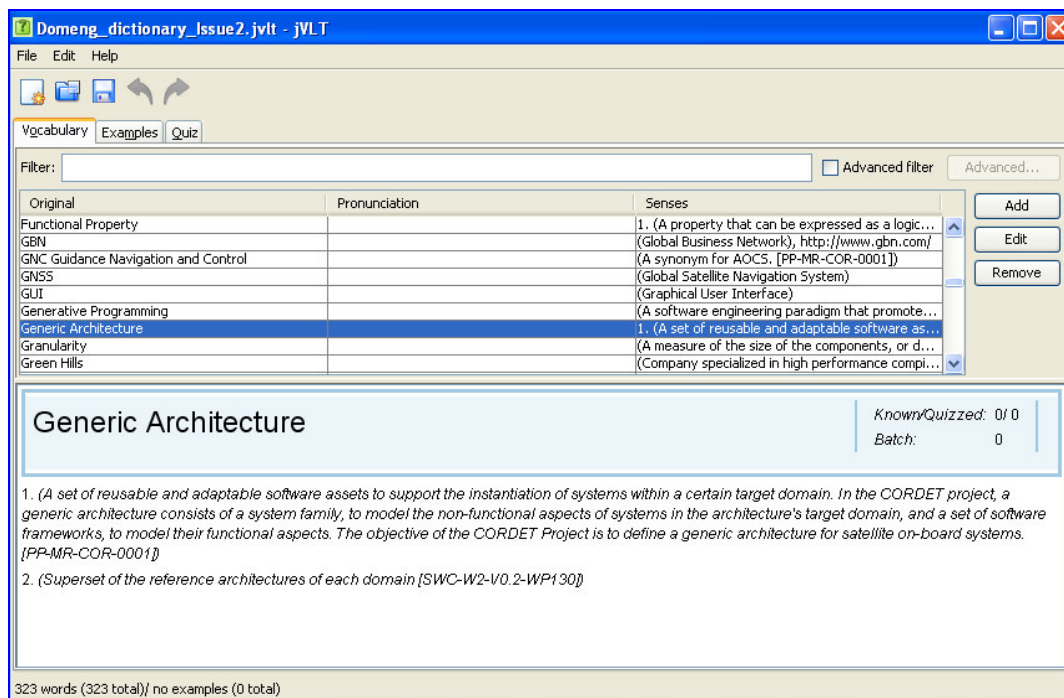


Figure 4-10: jVLT – Vocabulary Learning Tool

4.4. MODELS VALIDATION

4.4.1. VALIDATION OF REQUIREMENTS, CONTEXT, INFORMATION AND OPERATIONAL MODELS

Topcased also let the user validate the models. The validation process checks that the models are compliant with the OCL ("Object Constraint Language") constraints defined in the UML Specification. So that, not corrected associations, erroneous links, and so on are checked.

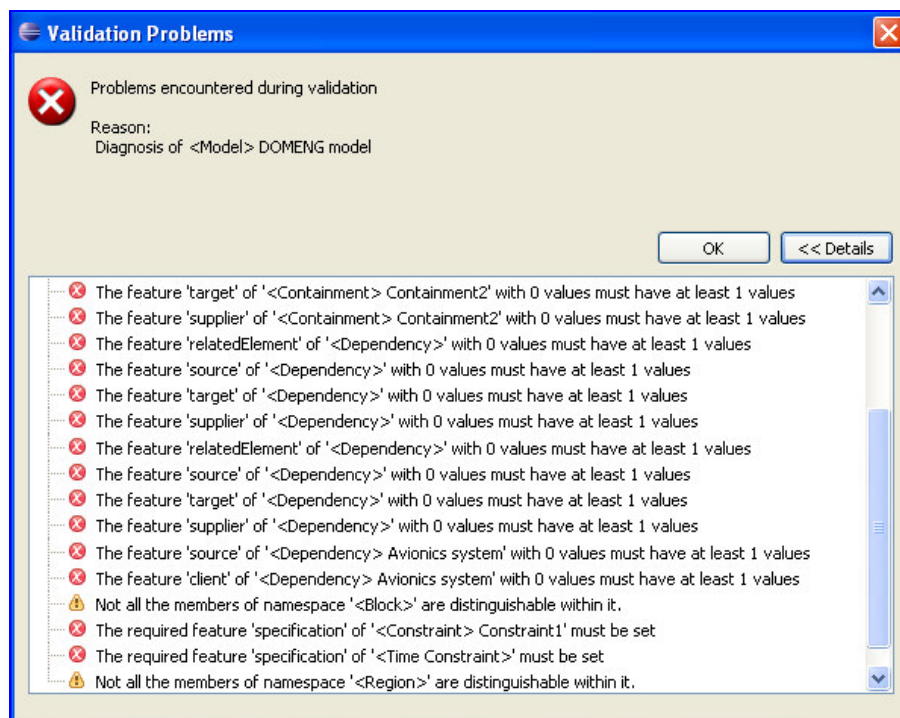


Figure 4-11: Unsuccessful validation of Requirements, Context and Domain Models

The above figure shows errors detected during the validation process of the domain models, whereas the below one represents the window shown when the validation is done successfully.

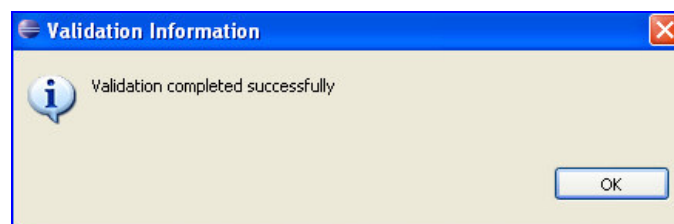


Figure 4-12: Successful validation of Requirements, Context and Domain Models

4.4.2. VALIDATION OF FEATURE MODELS

Xfeature tool provides two useful utilities:

- The application metamodel of the feature model can be extracted.

- Xfeature model can be validated to check inconsistencies. All application feature models are validated against their generated application metamodels.

Figure 4-13 shows the console messages generated when the metamodel has been created in a correct way. Figure 4-14 displays the console messages when the process of validation is successful.

```
<terminated> Run XFeature's 'generate' [Ant Build] C:\eclipse-SDK-3.3.1-win32\workspace\DOMENG_issue2\Space_Domain\Feature_model\xfbuild.xml

[echo] Application meta-model generated successfully.
[echo] No global constraints defined for this feature model ... skipping.
[echo] Application display model generated successfully.
[echo] All files generated successfully.
BUILD SUCCESSFUL
Total time: 21 seconds
```

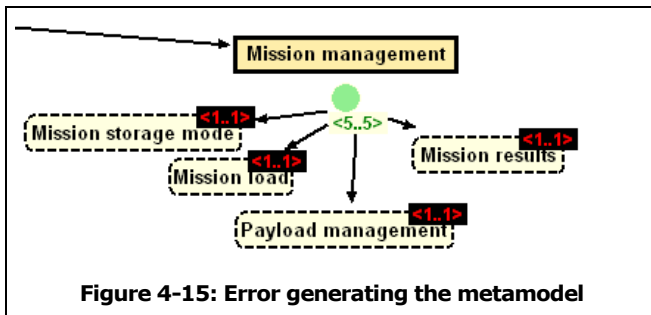
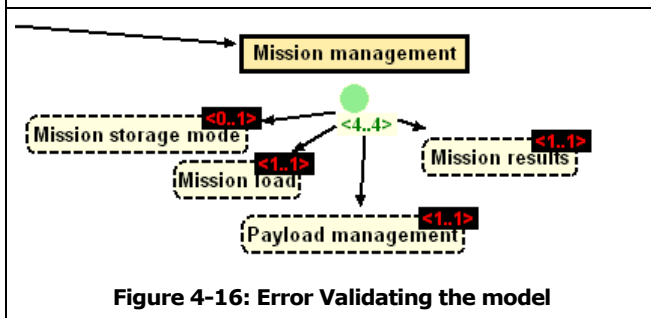
Figure 4-13: Successful generation of Feature Metamodel

```
<terminated> Run XFeature's 'validate' [Ant Build] C:\eclipse-SDK-3.3.1-win32\workspace\DOMENG_issue2\Space_Domain\Feature_model\xfbuild.xml

[echo] Model successfully validated against meta-model.
[echo] No global constraints defined for this feature model ... skipping.
BUILD SUCCESSFUL
Total time: 1 second
```

Figure 4-14: Successful validation of Feature model

These utilities facilitate the detection of inconsistencies in the design. Next figure compiles typical errors detected when the model is validated:

 <p>Figure 4-15: Error generating the metamodel</p>	<p>Error generating the metamodel</p> <p><i>ERROR: there are less child elements (4) of group 'GroupNode' than is the max cardinality (5) of the group.</i></p>
 <p>Figure 4-16: Error Validating the model</p>	<p>Error validating the model</p> <p><i>ERROR: The value '0' of attribute 'fm:cardMin' on element 'fm:FeatureCardinality' is not valid with respect to its type, 'oneOrMore'</i></p>

For instance in Figure 4-15 the group cardinality (number of features) is set to five but there are only four features. So, the error states that the cardinality value is wrong.

In Figure 4-16 the minimum feature cardinality (number of instances of that feature) is set to 0 and the minimum value is one. So the model cannot be validated.

Note: FD Configuration is used in all feature diagrams.

5. GUIDELINES FOR DOMAIN DESIGN

Once domain analysis phase has finished, the domain design can start. Its main objective is to specify the application architecture within a domain.

The best approach of all the proposed in CORDETs and DOMENG seems to be the CORDET-Toulouse one [Ref-DASIA-INTECS]:

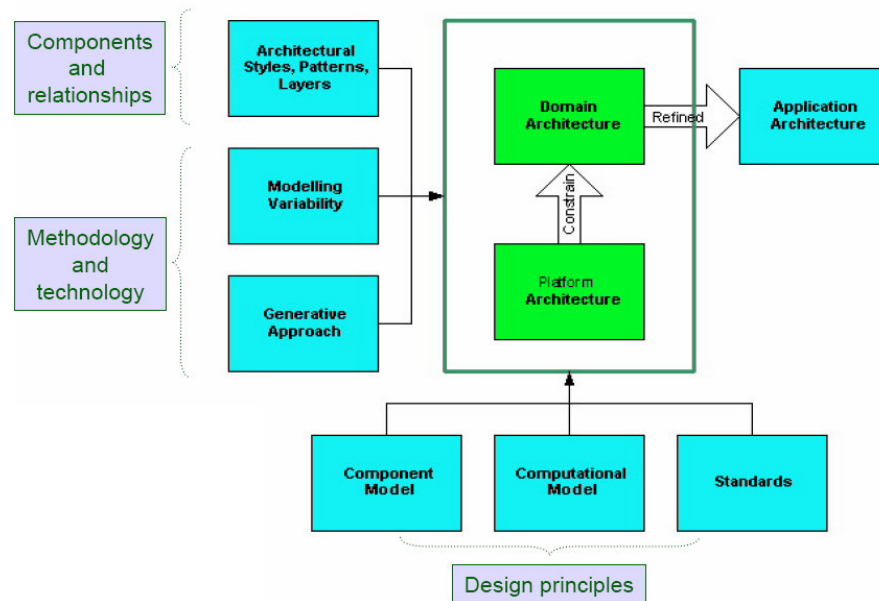


Figure 5-1: Approach to define reference architectures

The proposed approach in this methodology [Ref-CORDET-3] is based on:

- The architecture must accommodate the features, requirements and use cases captured by the domain analysis.
- It must be flexible enough to accommodate differences among individual applications to be built from the generic design.
- To design the architecture several concepts must be applied: architectural styles, architectural patterns, architectural layers, modeling variability, generative approach, component model, computational model, standards.

Therefore, applying this methodology a generic architecture for the space domain and the subsystems studied must be defined, providing different diagram models to define the architecture from different points of view.

6. CONCLUSIONS DEDUCED FROM THE TWO STUDIES ON THE DOMAIN ENGINEERING

The purpose of the domain engineering process is to develop and maintain domain models, domain architectures and assets for the domain. It includes the acquisition or development and maintenance of assets belonging to the domain. The domain engineering process can be defined as an organizational life cycle process and. This process has been instantiated by the Space Domain Engineering Data Acquisition followed by the Space Domain Analysis, which has been specifically focused on all software components embedded on board the avionics of science, service oriented and demonstration satellites.

Ground applications and software required for other sorts of missions and/or spacecrafts were excluded.

For programmatic reasons the scope of the product has been reduced to the space segment ([DOMENG]) of for satisfy any of three kinds of missions:

- Core missions – scientific/research oriented missions.
- Service oriented missions.
- Demonstration missions.

In addition the scope is explicitly restricted to satellites and excludes other space systems, like the ones for manned missions, landing probes, rovers, etc.

More specifically the scope of the domain under study is limited to the avionics of those satellites, thus other spacecraft relevant parts, like mechanical design, thermal design, etc. are explicitly excluded from the scope of the product.

The functions of the satellite avionics are manifold:

- Modes management
- Data handling
- Commanding and control
- Failure detection, isolation and recovery
- Mission management
- Power management
- Thermal management
- Attitude and orbit control
- Unit management
- Time management
- Communications

OBSW Application as part of general requirements for the domain grouped in the following categories:

- Architecture: The High-level structure of the system. The domain architecture is a generic based on the domain model, which satisfies the requirements.
- Applications: Mission specific parts of the on-board software.

- BB Definition: Building blocks (BB) are the shared (HW and SW) assets from which system can be built. The target system is constructed by assembling the adapted building blocks offered by the framework.
- Standardisation: New/adapted standardisation activities and establishment of a standard
- Technologies: The technological aspects of the space domain.
- Development: Process New/adapted phases, workflows, activities and artifacts, which describe the organisation of the development, project based on BB, standards and Reference architectures.
- Business models: New/adapted processes in the Stakeholders business

The tracks followed by DOMENG and COrDeT are quite similar. Both analyse at high-level the entire space domain and in a more exhaustive way a specific subsystem: the HMS for DOMENG and DH and AOCS for COrDeT.

During the design of the models of the domain analysis, COrDeT has included the use of MindMap tool to perform a hierarchical decomposition of the domain. But it does not provide the operational and informational models. DOMENG concludes that the operational and informational models are difficult to be applied when the domain studied is very broad. That is the case for the global space domain. But when a single subsystem is analysed these diagrams can be applied.

Both projects has encounter some difficulties using Topcased tool. Many commercial tools support SysML but Topcased is a free tool.

XFeature is selected for the design of feature diagrams with FD configuration. Domain dictionary is stored in plain text is COrDeT, whereas DOMENG also uses jVLT-Vocabulary Learning Tool (Version 1.0.2) to facilitate searches, stores and so on.

The high-level reference architectures proposed share some commonalities, such as they are component-oriented. But the DOMENG one includes two levels of schedulability, and the functional components are mapped to schedulability units that uses a SW assets library.

The most useful outputs of the studies for future work are:

- The domain engineering methodology
- A first iteration of the entire Domain Engineering Process. It serves as a reference framework for future studies. Models and results can be refined in next iteration.
- Propose high-level architectures to the global space domain.

Next steps focus on the definition of a transformation process from Domain Analysis to the Domain Design. Up to know, models define the context, features, requirements and behaviour of the system but the propose generic reference architecture must fulfill all these restrictions. Therefore, an important field of study deals with the verification of the fulfillment of the requirements defined during the domain analysis phase.

As well, this high-level architecture together with the complete first iteration serves as input to a second iteration to refine the process and focus on those parts more relevant for the space domain.

The current situation prevents the elaboration of the complete SFMECA tables in DOMENG, but not the preliminary steps.

A set of functional safety requirements as prevention and compensation mechanisms for HMS requirements are presented in DOMENG together with possible architectural mechanisms to prevent and/or compensate/recover from failures.