COrDeT Cannes : Use of domain engineering process to develop reusable architectures and building-blocks

G. Garcia[1], X. Olive[1], A. Pasetti[2], O. Rohlik[2], T. Vardanega[3], A.-I. Rodríguez-Rodríguez[4]
A. Stragapede[5], A. Jung[5]

1 : Thales Alenia Space, 100 bld du Midi – BP 99 06156 Cannes La Bocca Cedex (France)
2 : P&P Software GmbH, High Tech Center 1,Taegerwilen – Thurgau, CH-8274 (Switzerland)
3 : University of Padua, Dept. of Pure and Applied Mathematics, Via Trieste 63, I-35121 Padova (Italy)
4 : GMV Aerospace and defense, Isaac Newton 11 P.T.M., Tres Cantos, 28760 Madrid (Spain)
5 : ESA ESTEC : Keplerlaan 1, Postbus 299, 2200 AG Noordwijk (The Netherlands)

Abstract: This paper describes the activities performed in the frame of the COrDeT (Component Oriented Development Techniques) project performed in 2007-2008 by the COrDeT Cannes consortium (Thales Alenia Space, P&P Software, University of Padua, GMV Aerospace and defense) under ESA funding. This project aims to explore the usefulness of the domain engineering process standardised by ISO for the analysis, design and development of reusable software assets (both at architectural and building block levels).

Keywords: Domain engineering, Component, Model, Frameworks, Reference architectures

1. Introduction

The COrDeT Cannes project has been initiated and funded by ESA under the contract 20463/06/NL/JD. This study has the following objectives:

§ To define a process for product family-based development activities at both system- and software-level together with the profiles and tools required to support its application in an industrial context.
§ To identify and to define at the level of their functional and non-functional interfaces the product families covering the on-board satellite domain at both system- and software-level.
§ To demonstrate the proposed process and architectural concept by instantiating a subset of the product families to build an end-to-end demonstrator of an on-board system.
§ To get feedback from the space community in order to reach the largest agreement as possible on the methods and the products of the study through workshops.

The product families defined in this study will be referred to collectively as the COrDeT Product Families.

2. The domain engineering process

Domain engineering [1] is the process of analysis, specification, and implementation of software assets in a domain which is used in the development of multiple software products and as such the requirements and recommendations of ISO guides [2] its execution providing a rigours and structured process to organize, correlate and conduct the activities.

The Domain Engineering Process is divided into three phases: Domain Analysis, Domain Design and

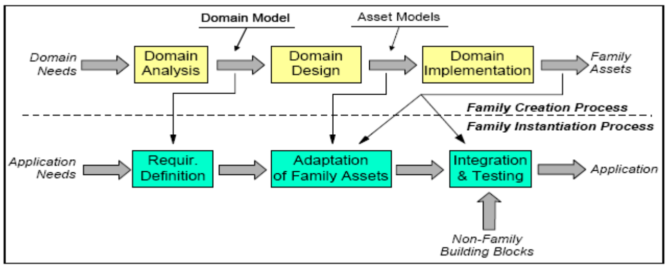


Figure 1 : Development Process for Software Product Families

2.1 Domain Analysis

The Domain Analysis phase defines the scope of the domain providing a set of models which represent formally all the knowledge obtained about the domain.

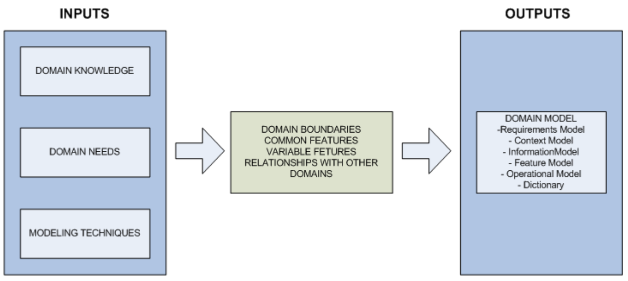


Figure 2 : Domain Analysis process

2.2 Domain Design

The Domain Design phase is used to specify the application architecture within a domain.

§ The domain architecture is a generic architecture based on the domain model. This architecture has to be documented and evaluated.

§ The software architecture represents the structured used during the modelling process. The architecture may be described at different level of abstraction showing components used to build the system, their relations, interfaces, etc.

A generic architecture is platform and code language independent. It's like PIM element in MDA approach.

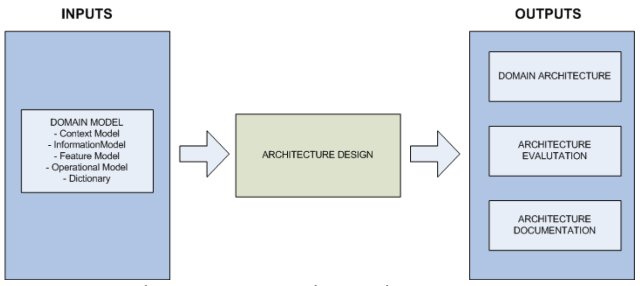


Figure 3 : Domain Design process

2.3 Domain Implementation Process

Domain Implementation starts with the identification of reusable assets. This identification is based on the domain model and the general architecture obtained in the previous phases. Using compilers and code standards the assets are implemented. Finally, reusable assets are created and saved in an assets' library.

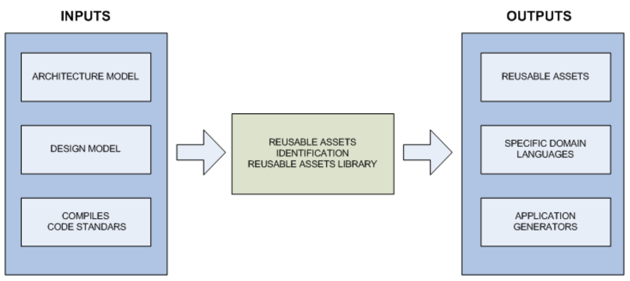


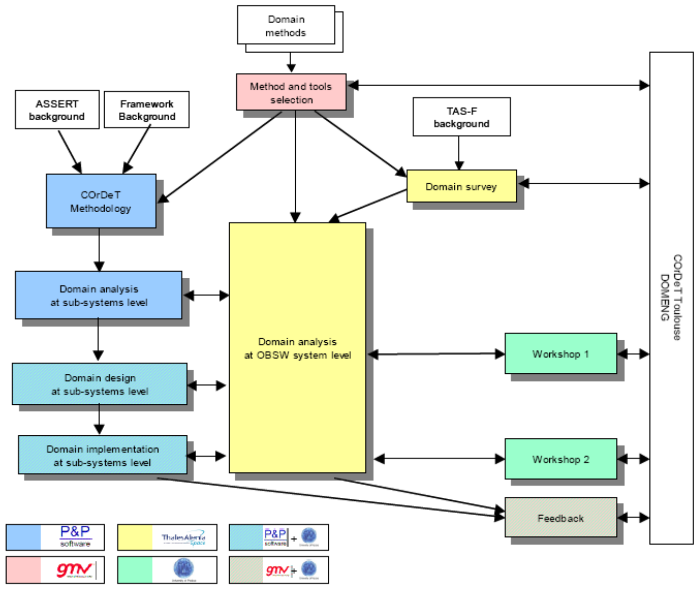
Figure 4 : Domain Implementation Process

2. The two COrDeT Cannes tracks

In order to evaluation the complete process from domain acquisition to domain implementation and to produce valuable results as well as a complete process implementation in the study time frame, the study has been split into two coordinated and closely coupled tracks :

§ An "Horizontal track" aiming to cover the whole space OBSW domain but limited to domain analysis

An "Vertical track" aiming to cover the whole domain engineering process ensuring its feasibility and proposing solutions for design and implementation of generic architectures and building-blocks



Domain methods
Method and tools selection
ASSERT background
Framework Background
TAS-F background
COrDeT Methodology
Domain survey
Domain analysis at sub-systems level
Domain analysis at OBSW system level
Domain design at sub-systems level
Domain implementation at sub-systems level
Workshop 1
Workshop 2
Feedback
COrDeT Toulouse DOMENG
P&P software
gmv

Figure 5 : COrDET Cannes Study logic

2.3 Horizontal track

2.3.1 Methodology

This track is mainly focussed on the domain analysis. The main objectives of this phase are to identify commonalities and variabilities across the studied domain and to capture them into a domain model.

The selected technology for capturing domain model is SysML, but due to ergonomic limitation of the current SysML modelers (traditional difficulties : to quickly refactoring a model during iterative design, to easy navigate though the model and understand it, …), a new approach has been adopted. In stead of using the SysML modeller to enter the model, we deduce the model from an automatic translation from a Mindmap[1] model to a SysML model.

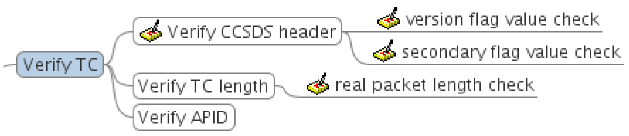


Figure 6 : Extract of the mindmap model

[1] http://en.wikipedia.org/wiki/Mind_map

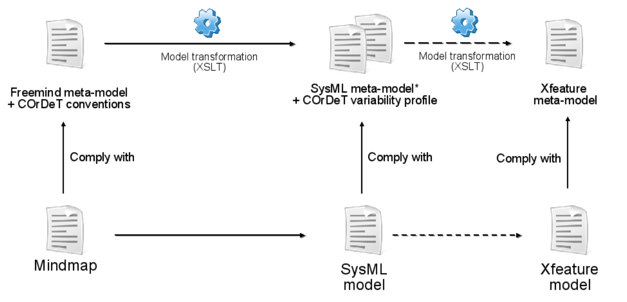

Model transformation (XSLT)
Model transformation (XSLT)
Freemind meta-model + COrDeT conventions
SysML meta-model* + COrDeT variability profile
Xfeature meta-model
Comply with
Comply with
Comply with
Mindmap
SysML model
Xfeature model

Figure 7 : Model transformations

2.3.1 Variability profile

In order to express variability directly in the SysML model, we have developed a profile called the Variability profile.

The profile permit to identify variation point and optional use cases or requirements. Each variation point has a attribute called Variability factor. Variability factors identified so far1 :

§ ProcessingModuleDependent
Variability linked to processor, processor I/O, …

§ AvionicsDependent
Variability linked to avionics equipments, network kind, topology

§ OperationalConceptDependent
Variability linked to the commandability and observability concepts, FDIR, …

§ MissionDependent
Variability linked to the mission itself like PL management, AOCS algorithms, …

2.3.1 Preliminary results

From the grouping of variability factors, and the reuse requirements, we are able to define a sketch of the OBSW architecture, consisting in a stack of the different variability factors.

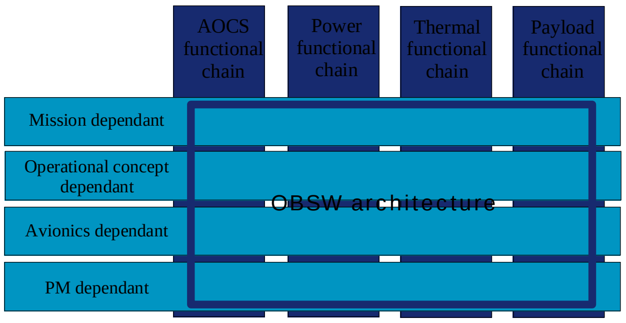


Figure 8 : Deducing the on-board architecture from tha variability factor analysys

This sketch permit to allocate the different requirements and use cases to each layer permitting to define interface level and contents of each building blocks.

2.4 Vertical track

This track applied the complete domain engineering process – from domain analysis down to the generation of source code – to a restricted sub-domain. The selected sub-domain was that of on-board applications. More specifically, the vertical track produced two *software frameworks* covering, respectively, the handling of PUS-compatible TM and TC packet (DH Framework) and the management of control activities (Control Framework).

The term "software framework" is used to designate a particular kind of product family where the 'product' to be built is a software application and the reusable assets from which it is built are software components embedded within a pre-defined architecture.

2.4.1 Development Process

The structure of the framework domain model is shown in figure 7. It consists of: (a) a domain dictionary that defines the terms used to specify the framework, (b) the shared properties that define the functional invariants to be guaranteed by the framework software components and their architecture, (c) the factors of variations that define the adaptation points where the framework components can be extended to accommodate application-specific behaviour, and (d) a feature model that captures the mutual dependencies among the factors of variations. The elements (a), (b) and (c) are defined using natural language whereas (d) is defined using feature modelling techniques and the XFeature tool [3].

The structure of the framework design models is shown in figure 8. The core of the framework design consists of UML2 models that comply with the FW Profile. The FW Profile is a UML2 profile that was specifically developed (in the ASSERT project) to support the definition of software frameworks.

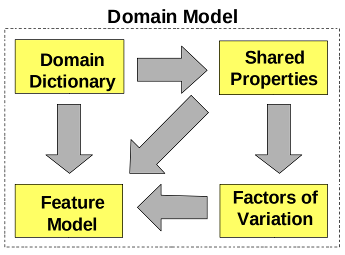


Figure 7 : Model transformations

The FW Profile is fully documented in [4]. The basic idea is twofold. On the one hand, the internal behaviour of the components is defined through state machines using a restricted version of the UML2 state machine model. The state machines implement the shared properties identified in the domain model. On the other hand, a controlled form of class extension through inheritance and object composition is used to implement the factors of variation identified in the domain model and to allow the framework-level components to be extended to implement application-specific behaviour.

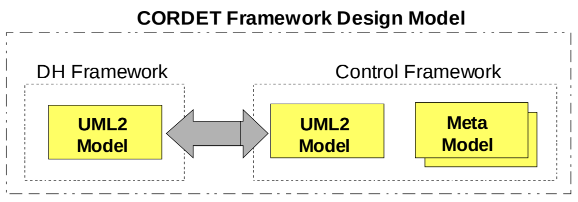


Figure 7 : Model transformations

In addition to the UML2 models, the design model of the frameworks also includes two meta-models to cover the definition of the data pool and on-board databases.

In the domain implementation phase, the design models are transformed into code. This is done using a code generator based on MOFScript technology.

2.4.2 Framework Target Domains

In general, the functionalities implemented in an on-board application can be divided into two broad categories. The first category comprises functionalities that are essentially sporadic and event-driven (where the “driving event” can be a command originating outside the application, a request for some information to be sent outside the application, a hardware interrupt, etc). The second category of functionalities are essentially periodic and consist of “activities” that, at every cycle, process the same set of inputs to produce the same set of outputs according to the same algorithm.

The DH Framework is aimed at functionalities of the first kind. The Control Framework is aimed at functionalities of the second kind. Thus, taken together, the two frameworks cover most of the functionalities present in an on-board application.

The two frameworks offer a set of components that implement the high-level functionalities of their respective domains. Consider the DH Framework first. The PUS defines the external interface of a DH application in terms of the *services* that the application must provide to other applications. The services are in turn defined in terms of *telecommand packets* that the application must be able to handle, and *telemetry packets* that the application must be able to generate.

The PUS implicitly defines the concept of an abstract telecommand packet and an abstract telemetry packet. This concept is independent of the particular service which the telecommand packet or telemetry packet supports. The definition of the abstract telecommand packets and abstract telemetry packets covers the features that are common to all PUS-compliant telecommand packets and PUS-compliant telemetry packets.

The DH Framework provides software interfaces and software components that support the implementation and manipulation at software level of abstract telecommand packets and abstract telemetry packets.

The DH Framework, in other words, transposes the PUS to the software level. The PUS standardizes the services to be provided by a DH application. The DH Framework standardizes the software interfaces through which those services are accessed at software level within a DH application.

Consider now the Control Framework. Its designed is based on the needs of the AOCS subsystem that is regarded as the prototypical on-board control application. The core of an AOCS application is the implementation of transfer functions that transform measurements from a set of sensors into commands for a set of actuators. Such transfer functions are implemented as digital filters that are characterized by a set of inputs, a set of outputs, an internal state, and an algorithm to compute the next set of outputs from the latest set of inputs and the current internal state.

Peripheral functionalities that are often found in AOCS applications include:

- Ø Management of AOCS operational modes;
- Ø Management of the external sensors and actuators;
- Ø Implementation of failure detection and isolation checks and recovery actions (FDIR);
- Ø Execution of AOCS-specific telecommands;
- Ø Generation of AOCS-specific telemetry packets.

The Control Framework directly covers the management of the AOCS operational mode through the operation mode concept and the activity manager concept.

The Control Framework does not cover the management of the external sensors and actuators. This is due to the fact that the interfaces of sensors and actuators are neither standardized nor do they exhibit any significant commonalities in existing missions. Standardization of interfaces to external units (not just for the AOCS subsystem) is possible but this is done at the bus interface level, not at the functional level. Such functionalities are therefore not specific to the AOCS and are best left out of a framework targeted at control applications.

The FDIR functionalities are not directly included in the Control Framework in its present form. It is believed that such functionalities present sufficient commonalities to be implemented in reusable and adaptable component and they might be included in a future release of the Control Framework.

The management of the telecommands and telemetry is not directly included in the Control Framework. However, the DH Framework is interoperable with the Control Framework and hence these functionalities are supported by the two frameworks taken together.

2.4.3 Real-Time Aspects

In keeping with the approach developed in ASSERT, the software framework only cover the application level of an on-board system and they only cover functional aspects. Non-functional – and in particular real-time – aspects are relegated to an RCM-based virtual machine. In practice, this means that the functional components provided by the framework are embedded within non-functional containers that endow them with real-time properties (such as an own thread or implementation of mutual exclusion mechanisms).

2.4.4 Framework Outputs

In concrete terms, the two CORDET Frameworks take the form of a set of components defined at both model and source code level. The components consists of UML2 classes that present a number of either abstract or virtual methods that application designers are supposed to implement or override.

Users can either take over the models and generate their own code from them or they can take over the source code and use it directly in their target applications.

In order to test both the models and the code, a small but representative application was built that demonstrates how the framework components can be extended to create application-level components.

All the results of the framework development activities are available on-line as free and open software from the project web site [4].

5. Conclusion

At the time of writing of this paper, the COrDeT Cannes study is not completed (we have just completed the domain analysis) and the results presented in this abstract are only partial ones, in the final version of the paper more details about concrete results will be given as well as a return on experience on the process.

The domain engineering process address the need of formalisation of the activities linked to the creation of software families that have been performed since years by the different actors but always in an empirical way.

This systematic way of exploring communalities and variability, should permit to define architectures and reusable building blocks definitions more robust to the reuse and evolutions.

Please find the last news and outputs of the project on the COrDeT-Cannes web-site :
http://www.pnp-software.com/cordet

7. References

[1] Domain Engineering, Software Engineering Institute. Carnegie Mellon http://www.sei.cmu.edu/domain-engineering/index.html

[2] 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

[3] The XFeature Tool, http://www.pnp-software.com/XFeature

[4] The CORDET-Cannes Web Site: http://www.pnp-software.com/cordet

8. Glossary

ASSERT : Automated proof-based System and Software Engineering for Real-Time applications

ECSS : European Cooperation for Space Standardisation

CORDET : Component Oriented Development Techniques

DOMENG : Framework for DOMain ENGineering

ESA :European Space Agency

ESTEC : European Space Research and Technology Centre