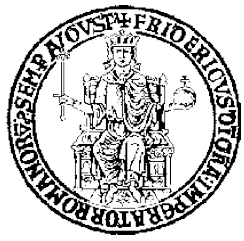
Università degli Studi di Napoli Federico II

Scuola Politecnica e delle Scienze di Base

Corso di Laurea Magistrale in Ingegneria Informatica

Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione



Tesi di laurea

Model Driven Engineering of railway control systems with the openETCS process

Relatore:
Ch.mo Prof. Stefano Russo

Correlatori:
Ch.mo Dr. Domenico Di Leo
Ch.mo Ing. Baseliyos Jacob, DB Netz (D)

Candidato:
Giovanni Trotta
matr.M63/0307

ANNO ACCADEMICO 2013 – 2014

Contents

List of Figures

Introduction

No one will ever think to build a house without having a clear and well-defined model. In every engineering branch, models represent the core of the development process and give a huge effort to clarify, specify, understand the project and predict the unknown throughout abstractions. On the other hand models within Software Engineering are really often unused, out-of-sync or play a second role. Mainly in this field more than others, the potential benefits are significantly greater.

Model Driven Development (MDD) [18] may make Software Engineering able to move an incredible step forward, and it can represent a generational leap in software development since the introduction of the compiler. A key method is the reliance on automation and the benefits that it brings.

However, like all new technologies, Model Driven Development shall be carefully analyzed and then introduced into the existing development processes. The number of companies that are using MDD is continuously increasing, but very few of them are using MDD to develop safety-critical embedded systems.

The goal of this thesis is to analyze the development process of a Railway safety-critical system using a Model Driven approach and showing weakness and strength points. Starting from the CENELEC EN50128 [10] standard constraints for Railway applications - Communication, Signalling and processing systems there are described the methodologies and the decision to adopt after each step of the development lifecycle.

Moreover, following the needs and the rules imposed by a project named: “OpenETCS” it is showed a case study of an “European Train Control System” (ETCS) [29] sub-system

deployment. The OpenETCS project purpose is to develop an integrated modeling, development, validation and testing framework for leveraging the cost-efficient and reliable implementation of ETCS.

The project aim is to provide a formal model of the ETCS System, the source code of ETCS software and a tool chain necessary to develop and verify them. Moreover, it will use "Open Standards" at all levels, including hardware and software specification, interface definition, design tools, verification and validation procedures and last but not least embedded control software.

In most cases the reason for this is not only to reduce the costs but also to reduce the binding and strong dependence on the software producer. This is especially essential for technical software in the railway business, where equipment has been used for several decades and long-term support has to be ensured, even if the original software producer exits the market for some reason.

Last but not least this thesis presents also a brief description of the Agile Method Scrum [39], its advantages and disadvantages and how it can be adapted to schedule the work for the development of a Safety-Critical systems (SIL 4) as it used in the OpenETCS process.

This project mainly addresses interoperability for the European railway sector, promoting competitiveness of the European industry giving them a leading edge in the software service industry by applying a new concept called 'Open Proofs' for the first time, fulfilling safety and security demands as requested by the EU Parliament resolution A5-0264/20013. A key element for improving that situation seems to be a greater degree of standardization in particular for the ETCS on-board equipment on various levels. Standardization by applying open source licensing concepts is the focus of this project .

Starting from the requirements engineering, passing through modeling, coding, verification and validation activity, the rules and policies imposed by the standards, committers, and markets in order to improve the quality and maintain the cost low will be analyzed step-by-step.

For every phase, a tool benchmarking based on the process needs was performed show-

ing all the possible choices. To have a clear view of the process and the project, a case study shows the Model Driven Development Approach combined with the Agile Method Scrum and the CENELEC V-Model. To develop a sub-system of the European Train Control System Software that manages train position and related information. This module named: "Balise Group Orientation" (BGO) shall analyze the current running direction of the train considering data from external devices.

Analyzing the requirement, specified in the System Requirement Specification SUBSET 026 [42], is the first step in the development. These requirements shall be maintained and forwarded to the other phases with a well-defined traceability. Drawing Models is the core phase of the project. These models can be classified in two different types. The first type are the ones that use abstraction to describe the system. The second type are implementation models that use an embedded language to develop the system. Verification and Validation represent a key and crucial aspect of the process, this phase interact with the others to improve the quality.

This case study and part of the thesis, have been developed during a six-mouth internship at Deutsche Bahn Netz AG in Munich Germany where the author had the chance to have really hands on the process.

In addition, part of this thesis takes into account the automatic generator of source code, which represents the key benefit of the Model Driven Architecture.

The **first chapter** describes the domain problem of the ETCS environment, listing the main aspects of the Train Control Systems as much as the devices that interact with the on-board unit software and also the protocols used.

The **second chapter** lists the rules to follow in order to develop a Model-Driven software engineering process for Railway Control System making references to Railway and safety-critical system standards, the model driven architecture standard, tools and methodologies to use.

The **third chapter** is focused on the OpenETCS project and its process; it considers the way to approach and resolve the problems, the decisions taken to avoid ambiguity and

antithetical points imposed by different standards and methods.

Chapter four is a brief description of the case study. Organize and store requirements to have a full traceability, passing from an architectural model to an implementation model, to simulate system execution and to generate test cases for testing, are all activity taken into account in this part.

The **Last chapter** shows the weaknesses and straights of the used Model Driven Engineering process and more specifically inside the OpenETCS project. Moreover, an evaluation and tools benchmarking shows the significant issues of any software to support the development process.

There is, also, a look at the future by listing all the possible improvements of the approach, methods and processes.

Chapter 1

ETCS/ERTMS

1.1 Introduction

ETCS/ERTMS is an interoperability system for railway signalling. It has been required by the European Commission not only to achieve efficient cross-boarding working, but also to allow trains equipped by one supplier to work on infrastructures equipped by another supplier [24].

The European Rail Traffic Management System (ERTMS) is the concept of a new generation of train control and signaling for European railway traffic control of inter-operable systems. The European Train Control System (ETCS) is the hardware and software, which needs to be developed to put the concepts of ERTMS into operation. The ETCS technology has different levels (Level 0, Level 1, Level 2/3), which offer different combinations of capacity and performance. This covers a lot of types of railway control system and different technical platforms, together with a good understanding of needs for the various sub-system serving railway operation and train control to interface and work together. ETCS presents a completely new situation in that the constituent parts are independent, with common data definitions, message structures and processes. The need for integration of sub-systems, previously regarded as stand-alone, imposes restriction on sub-systems engineers who must ensure that any proposal for functional on engineering

change must be considered in the complex total system [29].

The work required the definition of a set of ETCS requirements, sub-systems boundaries, interfaces and operating procedures that shall be applied and accepted by the European Union Commission.

1.2 ETCS Requirements

ETCS shall be a future standard for the main railway lines in Europe in order to reduce technical and economic risks by following the European Commission rules. A classification of three sub-sections of economic, technical/operational and political requirements are commonly considered [29].

Economic Requirements

ETCS has a very important impact on the European Union economy by opening the cross-border traffic. Private operators, in fact, have an high interest in using ERTMS solutions to speed-up their process and become more efficient.

Nonetheless, Life Cycle costs represent a capital investment for railways companies. Initial investment, operational costs, maintenance and replacement costs must be taken into account for high number of vehicles and corridors [29].

Technical/Operational Requirements

ETCS increase the safety in some European countries by using standardized platform.

Maintainability is one of the main aspects of the ETCS environment since it has a long Life Cycle as well. The removal of redundant track equipment reduces maintenance, the change of failures and increases availability of the total system.

The architecture of ERTMS/ETCS system shall be implemented depending upon suppliers, the layout installation, the quantity of sub-systems and the place where to implement them, and on the interfaces caused by overall the environment [29].

Political Requirements

The European Community imposes to have a common standard between countries and no one of them may have a different approach to in compliance with the European Commission rules.

All the legacy systems shall be replaced by ETCS and a speed of migration is needed differentiating itself for the trackside equipment and the on-board equipment [29].

1.3 System description

The ETCS standard is composed mainly by two parts: The **ETCS Track-Side** equipment and the **ETCS On-Board Unit** equipment, the so called On-Board Unit (OBU).

The track side equipment is composed by three kind of devices: Eurobalise, Euroloop and Radio RBC that exchanges information with the running trains over the track.

The standard controller inside the train is basically a device that inputs, filters and computes all the information received by the track. It is composed of a Kernel, an "odometry" mechanism and other parts that interface with the Trackside Equipment and the Driver of the Train. The Driver of an ETCS equipped train has two control panels by which all ERTMS information is received and actions are taken. These are Driver Radio Interface (DRI) and the Driver Machine Interface (DMI). The DRI announces incoming calls or interrupts existing calls with emergency messages. Instead, with a DMI, a driver can input data and change the status during the journey.

ETCS Control System, its parts and interfaces between components are shown the entire in Figure 1.1. For each component in the following chapter are explained the functionality and the scope. Ideally, the ETCS can be divided into two macro parts: the ETCS On board and the ETCS Trackside. The first one is completely located on the train and interact not only with the Trackside equipment such as Eurobalises, Euroloop and Euroradio but also with the National System. ERTMS/ETCS trains use the national system's Specific Transmission Module (STM) installed on-board with which the equipment can provide standard informations taken from the ground-based national systems.

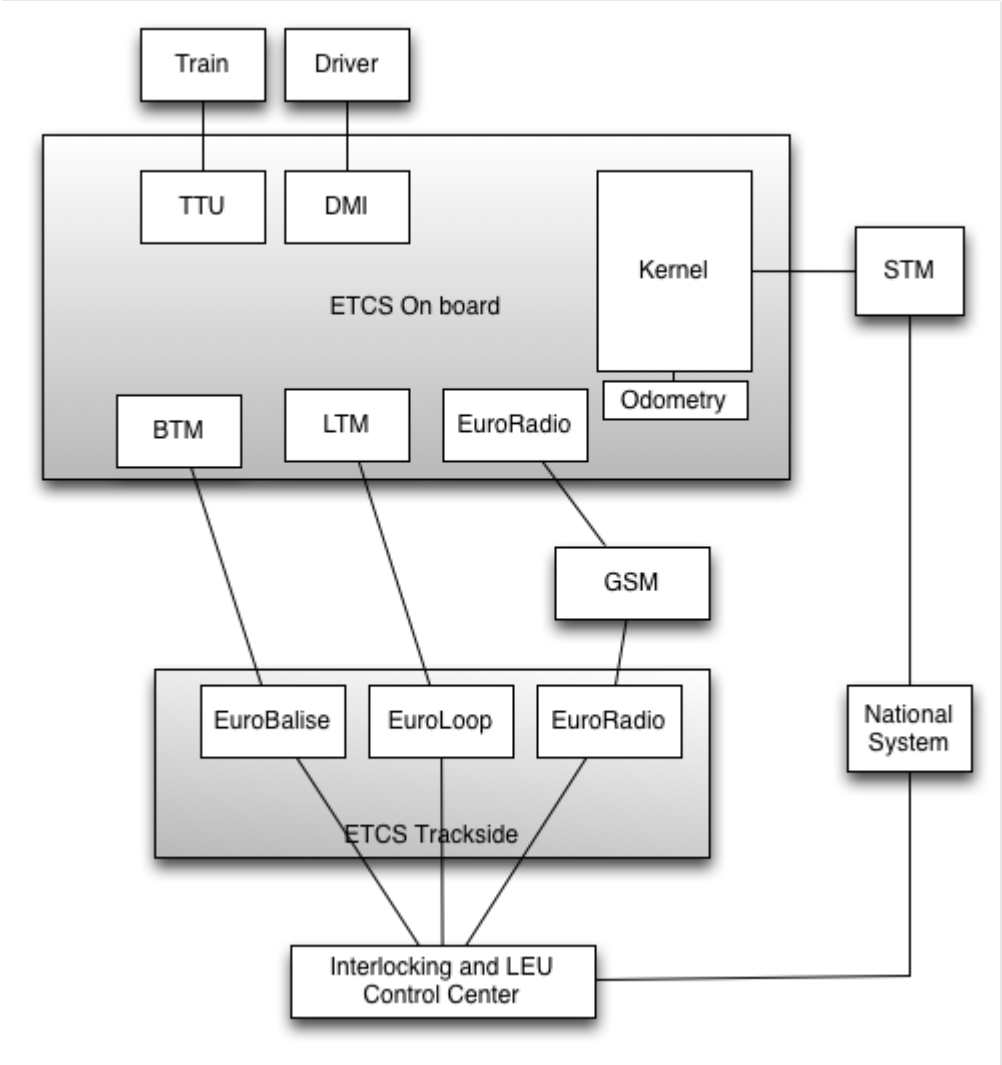


Figure 1.1: ETCS System

- **Train** Train equipped with ETCS/ERTMS Control System
- **Driver** The train driver
- **DMI** Driver Machine Interfaces
- **Kernel** the core of the On board equipment
- **Odometry** A component that receives relative measures (i.e. speed) from sensors
- **STM** Interface of the ETCS On board equipment with the Nation System Authority
- **BTM** Interface module of the EuroBalises

- **LTM** Interface module of the EuroLoop
- **EuroRadio** Component that sends and receives critical information via GSM Radio.
- **EuroBalise** Trackside component that sends information to the running train
- **EuroLoop** Trackside component that sends information to the running train via interlocking
- **National System** Authority that allows or refuses permissions of movement
- **Interlocking and LEU Control Center** Control Center for the Interlocking,and signaling

The ERTMS/ETCS on board equipment is a computer-based system, composed of modules and units. These can be individual parts, or combined in a rack according to the implementation policy of each supplier. Kernel module is the container of the major logical functions of ETCS.

Train Interface Unit (TIU) interfaces between the train and ETCS **Balise Transmission Module (BTM)** interface between the ETCS OBU and Eurobalises installed on the trackside.

The **Loop Transmission Module (LTM)** interfaces between the ETCS OBU and Euroloops installed on the trackside.

The **Euroradio** module, interface between **GSM-R**, which is a radio communication network that is used for bi-directional exchange of messages, and ETCS OBU The **Odometry** module includes a processing part and different sensors, to estimate the movement of the train.

GSM-R Global System for Mobile Communications-Railway is a wireless communication standard for railway control and applications. The European Rail Traffic Management System uses it for communications between train and railway, regulating traffic and

movement by the use of a control center. It guarantees performance at speeds up to 500 Km/h without connection loss. When the train passes over Eurobalises, it transmits its new position and its speed, then it receives back agreement or disagreement to access the next session of the track and the movement regulations.

A Juridical Recording Unit (JRU) that memorized events for legal investigations in case of accidents. A specific **Transmission Module (STM)** interfaces between the OBU and the National System Authority a driver **Machine Interface (DMI).**

The switches and signals of a railway are distributed geographically over a considerable area in the vicinity of a station. In railway, an **interlocking** is an arrangement of controlling signals that prevents conflicting movements through an arrangement of tracks such as rail junctions or crossings. Interlocking is designed so that is not possible proceed in the movement unless the route has been proven safe.

1.4 ETCS Trackside

1.4.1 EuroBalise

Eurobalises are devices embedded on the tracks that transmit several information to the running trains. Eurobalises are organized in groups. Each group is composed of one up to eight Eurobalises. Every single Eurobalise is able to send one message to the running train. Running trains collect, check the consistency and collapse all the messages of the group[42]. Transmission system is composed of an antenna unit and the EuroBalise Transmission Module function. The use of Eurobalises is allowed in all the Layers of the ETCS/ERTMS. Each component of the group has an internal number (from 1 until 8) and an identification number of the group. Every Eurobalise group has its own co-ordinate system, to understand the direction of the running train [29].

1.4.2 EuroLoop

EuroLoop is an extension of the Eurobalise over a particular distance which basically allows data to be transmitted continuously to the vehicle over cables emitting electromagnetic waves. Euroloop is a coaxial cable with termination resistor at the end. The cable can be installed both on the inner and outer side of the rail. Euroloop has being specified to complement the Eurobalises, so that a common antenna can be used on-board to read both Eurobalise and Euroloop. This device typically transmits telegrams when a Eurobalise interrogation signal is detected in the loop cable, which means a train is present on the loop, and the transmitter is activated [29].

1.4.3 EuroRadio

The **Radio Block Center (RBC)** is a protocol radio transmitter controlled by National System, which gives information to the running train on GSM-R. In has a stack that interfaces be- tween the ETCS application and the radio for data transmission. Its role is to protect from threats when it is transmitted over a non-safe radio. ETCS specifications are only concerned with different implementations where interoperability is unaffected [29].

1.5 ETCS Levels

Levels in the ETCS environment interface different way of signaling and control trains.

Level 0 the driver is required to enter data such as maximum train speed at startup.

In this Level the train can respond to any level change instruction received from Eurobalise group. The DMI provides only a speedometer display. The speed maximum limit is entered by the Unfitted mode and the driver can decide the speed over the limit [42].

Level 1 (fig. figure 1.2) the driver is supported by line-side signaling and DMI informa-

tion. ETCS OBU will indicate the distance to the end of the current movement authority. Interlocking and Eurobalise regulate operations according with movement authority. A traffic light governed by Authority gives signal to the driver who shall stop the train waiting until the moving on signal [42].

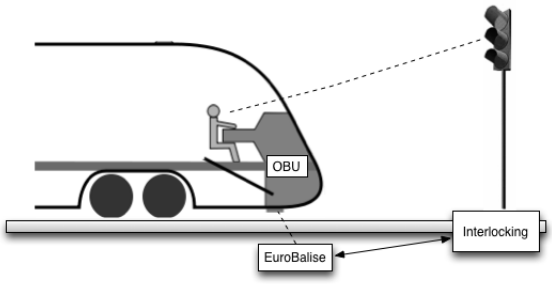


Figure 1.2: ETCS Level 1

Level 2/3 (fig. 1.3, 1.4) the driver could not observe line-side signals. Indeed, in most cases there will be nothing to observe. ETCS OBU will indicate the distance to the end of the current movement authority, itself enforce the maximum line speed and will generate a speed reduction envelope. Operating regulations are enforced through the action of the Interlocking and RBC [42].

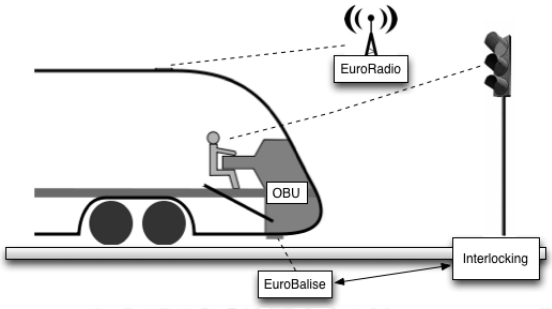


Figure 1.3: ETCS Level 2

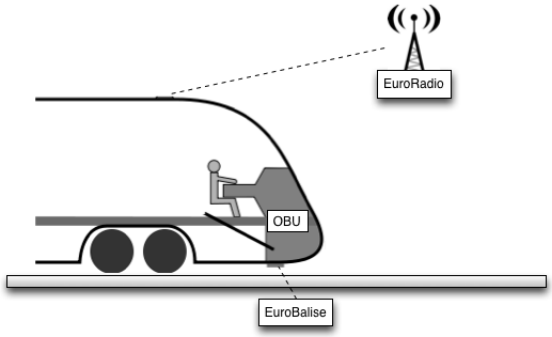


Figure 1.4: ETCS Level 3

1.5.1 Stack Protocol Layers

The ISO Open Systems Interconnection (OSI) imposes a protocol stack (fig. 1.5) that define seven well-known layers: Application Layer, Presentation Layer, Session Layer, Transport Layer, Network Layer, Data Layer and Physical Layer which is adopted by ETCS as well. An application layer is responsible for the transfer and the management of documents or messages that is common services to all application. An application, as far as EuroRadio RBC is concerned, refers to ETCS applications. There is a convention within the EuroRadio specifications: the interfaces describe vertical connection points between different layers, protocols define the horizontal interaction between different layers between, endpoint and layers. This means that interfaces are generally within an implementation, and are not mentioned by specifications [29].

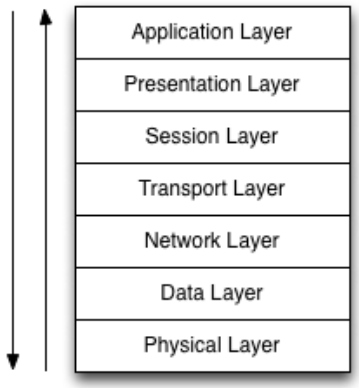


Figure 1.5: ETCS Protocol Stack

Chapter 2

Model-Driven Software Engineering for Railway Control Systems

The CENELEC (Comitè Europèen de Normalisation Electrotechnique) and the IEC International Electrotechnical Commission approved the Standard EN50128 for railway applications based on communication, signalling and processing system. This standard gives the procedures and requirements for the development of electronic systems in the railway domain but it is possible to use it in any safety environment. In addition, it supports a big part of electronic systems in a large range, from safety-critical system to the non- critical one, such as management information.

It supports the implementation and development of firmware, operating systems, tools and applications. Safety Integrity Level represents one of the main aspects of the standard. Five levels are explained, from Level 0 which describes the process and the requirements for non-safety system, until Level 4 for a high-required safety Integrity. Higher is the level, more dangerous can be the consequences of system failure [42].

A **Software Plan**, as shown in figure 2.1, is built to develop a process that describes a system based on flow diagrams. From a **System Requirements Specification (SRS)** document, developers shall **identify the Kernel functions,** they will create a **Software requirement Specification**, they will build the software, and for each function determine

the correspondent Software Integrity Level. The **Design Phase, development and verification testing** are based on a **Quality Assurance plan. Validation and Maintenance** are basically the last steps of this process. The whole process is iterative, whenever an error, a failure or an inconsistent state is discovered in every step, it shall be notified and forwarded.

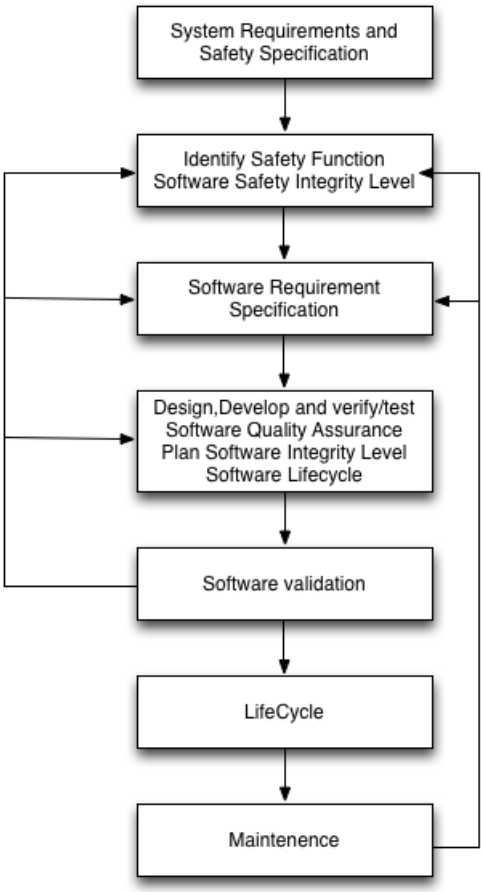


Figure 2.1: Illustrative software Route Map

2.1 Functional Safety

Safety is one of the first definition that shall be clarified in the process of developing a railway control system. This is the freedom of physical injury or of damage to the health of people, either directly or indirectly as a result of damage to property or to the environment.

Functional Safety is part of the overall safety that depends on a system or equipment operating correctly in response to its inputs. The international standard IEC 61508 (which is a basis for other standards like EN50128) , for Functional safety of electrical/ electronic/ programmable electronic safety related systems, provide a mean for users and regulators to gain confidence when using computer-based technology. The IEC 61508 is the international standard used by safety equipment manufacturers to verify and document that their products are suitable for use in a safety-critical environment.

2.1.1 Safety Process

Software Development strictly depends upon the Safety Integrity Level. Each level in fact specifies techniques and measures required at each stage. These techniques can be **Mandatory** (**M**), **Highly Recommended** (**HR**), **Recommended** (**R**), **Not Recommended** (**NR**) and **unspecified**.

A **Software Quality Assuranc**e Plan identifies, monitors and controls all the activities, which are necessary to achieve the required quality. It is strictly necessary to establish quality assurance a priori to have a clear life cycle model and allow verification and validation activities to be undertaken effectively. At the System Level a **Safety Plan** has been established. For each activities input, outputs and relevant criteria must be defined [10].

2.1.2 Safety Integrity Levels

The standard EN50128, using IEC 61508 as a base, specifies 4 levels of safety performance for a safety function. These are called Safety Integrity Levels.

Safety Integrity level 0 (SIL0) is the lowest level and Safety Integrity level 4 (SIL4) is the highest. The standard details the requirements necessary to achieve each safety integrity level. These requirements are more rigorous at higher levels in order to respect the safety integrity and to achieve the required threshold of dangerous failure.

The rate of unsafe failures can be measured numerically, every SIL has a target probability

of dangerous failures of a defined safety function and was originally intended for use when qualitative hazard analysis has been carried out and numerical risk values are not available, as in the case of software.

Safety Integrity Level	Probability of a dangerous failure per hour
4	$10^{-9} - 10^{-8}$
3	$10^{-8} - 10^{-7}$
2	$10^{-7} - 10^{-6}$
1	$10^{-6} - 10^{-5}$
0	-

Table 2.1: Safety Integrity Levels - Probability of failure

SILs are intended to provide targets for developers to accomplish complex systems and for complex software, whose failures are systematic and not random. Thus, SILs are used to define the rigor used in the development processes and to reduce the human factor [1]

It is never possible to prove safety and the complete safety cannot be achieved. For this reason, it is necessary to try to increase confidence as high as the Level required. Safety Engineers then have two goals: to achieve the required safety and to demonstrate the achievement.

2.2 Software standard process

2.2.1 Software Requirement Specification

The aim of the requirement analysis phase is to provide a comprehensive set of documents for each subsequent phase. A Requirement Manager take all the responsibility to write a Software Requirement Specification showing that it contains all the requirements written in a proper way following some properties. Functionality, robustness, maintainability, safety, efficiency, usability and portability are properties provided by each requirement [20].

Moreover a requirement shall be compete, clear, precise, unequivocal, verifiable, testable,

maintainable and feasible.4. Traceability documents allow the requirements to be traceable and linked. In validation stage every requirement has an important rule and it is required to demonstrate the validity of the system throughout all phases of the lifecycle [20].

A full traceability shall be maintained (fig. 2.2) overall the process passing through design, implementation, maintenance, verification testing and validation. The Software Test specification is an activity, which shall be separated by the other. Both **Software Requirement Specification** and **Overall Software Test Specification** are used as input for the Software Requirement Verification.

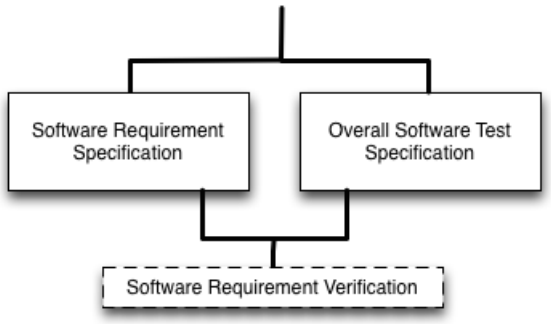


Figure 2.2: Requirement Phase

2.2.2 Software Design and Development

The Design phase has task of creating a Software Architecture Specification Document that achieves the requirement of the software. Even a Software design specification and an Interface Specification shall be done in that step. The **Software Architecture Specification,** shall identify all software components, the interaction between them and the interaction with the hardware. The **Software Design Specification** runs in parallel with the **Integration Test Specification**, It has the aim of clarifying the design of the component under construction. These two processes shall be used as input for the **Software Design Verification** (Figure 2.3) [10].

Every component shall be clear, independent, and identifiable within a management system. It has to cover a subset of requirements. Every component shall be comprehensi-

ble, verifiable, and maintainable. Basically a software Component has clear and precise functions, information flow exchanged between components, sequencing and time related information, concurrency, data structure and properties.

SIL 4 Highly Recommend is a modeling based on formal methods, a modular approach otherwise is Mandatory in order to have a clear view of the entire system and a precise approach to it. In the Modeling it is highly recommended a Data and Control Flow Diagram, finite state Machine Diagram, Sequence Diagram and other formal Diagrams. All these Diagrams describe the system and a hierarchy of parts [10].

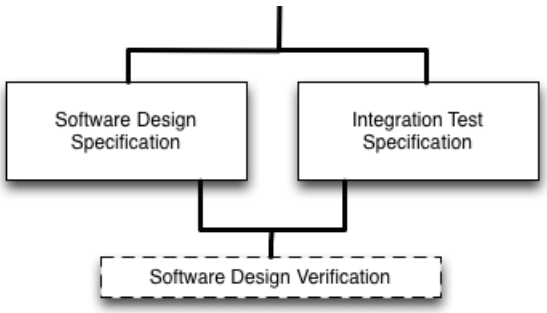


Figure 2.3: Design Phase

2.2.3 Testing, Verification and Validation

Starting from a test specification and software testing activities it is required to be sure that the behavior or performance correspond to what is required. Every Manager of the team (Requirement Manager, Designer, Implementer) shall write a document, which perform requirements and forward this document to Verifier.

The testing and Verification and Validation activities are categorized into two main parts**:** **Source Code Verification** and **Software Validation**. The first parts, as shown in figure 2.4, has the aim of verifying the software component under construction, and integrate it inside the main software. A Software Validation is followed in the process by a Software Deployment and Software Maintenance [10].

After the **Test Report** an appropriate measurement equipment shall be calibrated. Every Test shall be present in the Test Report and it shall be verified, documented and repeatable.

The Testing and Verification step is highly recommended to carry out the formal proof with static and dynamic analysis using metrics and traceability [10].

A **Test coverage** for code and functional black box testing is mandatory within the process. In spite of this, verification activities examine with a judgment based on evidence, the process, documentation, software or application of specific development phase. Moreover, it fulfills the requirements and plans with respect to completeness, correctness and consistency [10].

A **Report and a Verification Plan** must be output documents of that phase. In safety Integrity Layer 4 Software validation is the demonstration that the process and their outputs fit the requirement specification, the safety integrity level, and the Quality Plan [10].

The main activities of that phase is to demonstrate by analysis that all the software requirements are specified, implemented, tested and fulfilled with the safety restrictions.

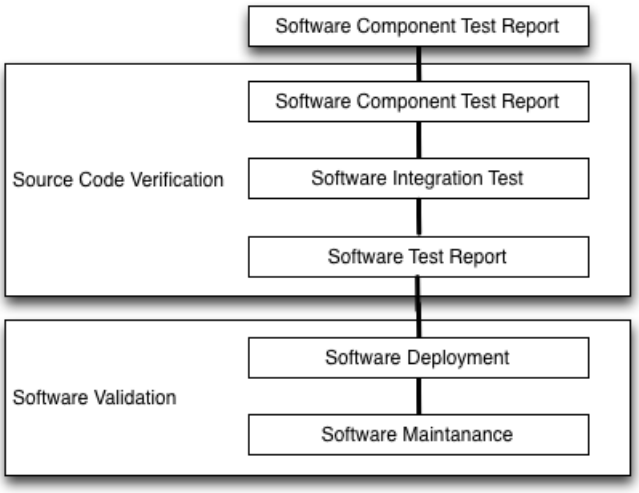


Figure 2.4: V&V Phase

2.2.4 V-Model

Every engineer of industrial project shall have well clear its phases and its goals. A conceptual model designed to produce a simple understanding of the complexity associated to system development across all the life cycle is the V-Model [10].

V-Model is a System Development life cycle (SDLC) based on an extension of the waterfall model. It has associated a testing phase for each stage of the developing and every stage starts only after the completion of the previous one. There are several different types of V-Model one of this is the German V-Model "Das V-Modell" approved by German government authority.

The name V derived from the graphical representation (Figure 2.5) and because the testing phase is planned in parallel. There are, in-fact, verification phase on one side and Validation phases on the other side. Coding joins the two sides of the V-Model. A real important goal of the V-Model is to record all the information pertinent of the software inside all the life-cycle phase.

V-Model Phases

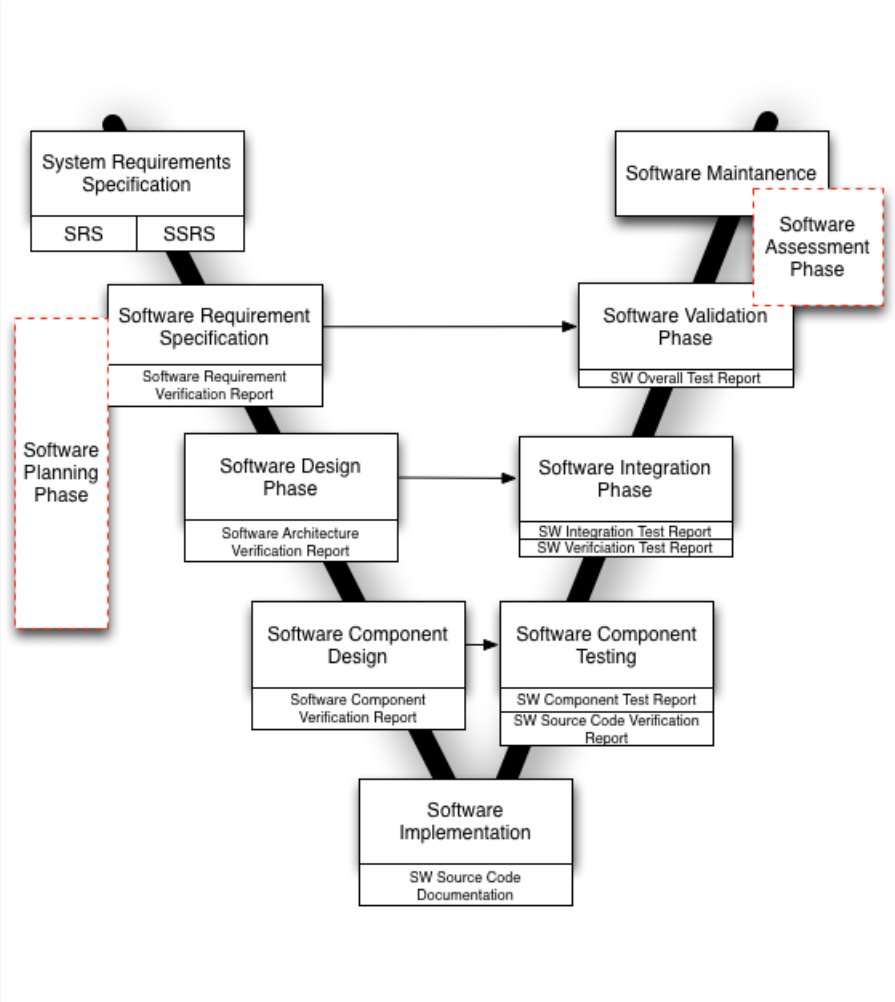


Figure 2.5: V-Model

The **Requirement Phase** is the first phase in the development cycle where the product requirements are analyzed. This involves a lot of activities to understand what to do and it is a very important activity that has to be managed very carefully.

The **Software Architecture and Design Phase** is the step dedicated to design that describes the system once they have a clear definition of the product requirements. In this phase Developers understand the hardware and how to setup communication for the product under development. The data transferred and communication between the internal modules and the outside world shall be clearly understood and defined in this stage.

The **Software Components Design Phase** is the phase where detailed internal design for all the system modules is specified. It makes the system architecture compatible with the other modules in the system architecture.

The **Coding Phase** is the actual coding activity of the system modules designed in the design phase. The coding is performed based on the coding guidelines and the Development Standards. The code goes through numerous code reviews and is optimized for the best performance before the final build.

The **Software Components Testing Phase** has the aim of validating the unit tests specified in the design phase. It helps to eliminate early stage bugs and defects.

The **Software Integration Phase** is associated with the architectural design phase and checks the entire system functionality and the communication between systems under development.

The **Acceptance Testing or Software Validation Phase** is associated with requirement analysis. A testing of the product in the user environment shall be done, taking into account all the requirements of the component.

The **Acceptance Test** uncovers the compatibility issues with the other systems available in the user environment.

2.2.5 Organization, roles and responsibilities

Safety Integrity Levels, as they are described in CENELEC EN50128, specify the target level of risk reduction. All the implemented software shall be specified as one of the five levels (from SIL 0 to SIL 4). Organizations, roles and responsibilities are strictly defined for each SIL (figure 2.6) in the standard referring to software management and organization. In order to ensure quality all the personnel that has responsibilities for software is organized, empowered and capable of fulfilling its responsibilities. The entire group of employees that is involved in the project shall be named and recorded [10].

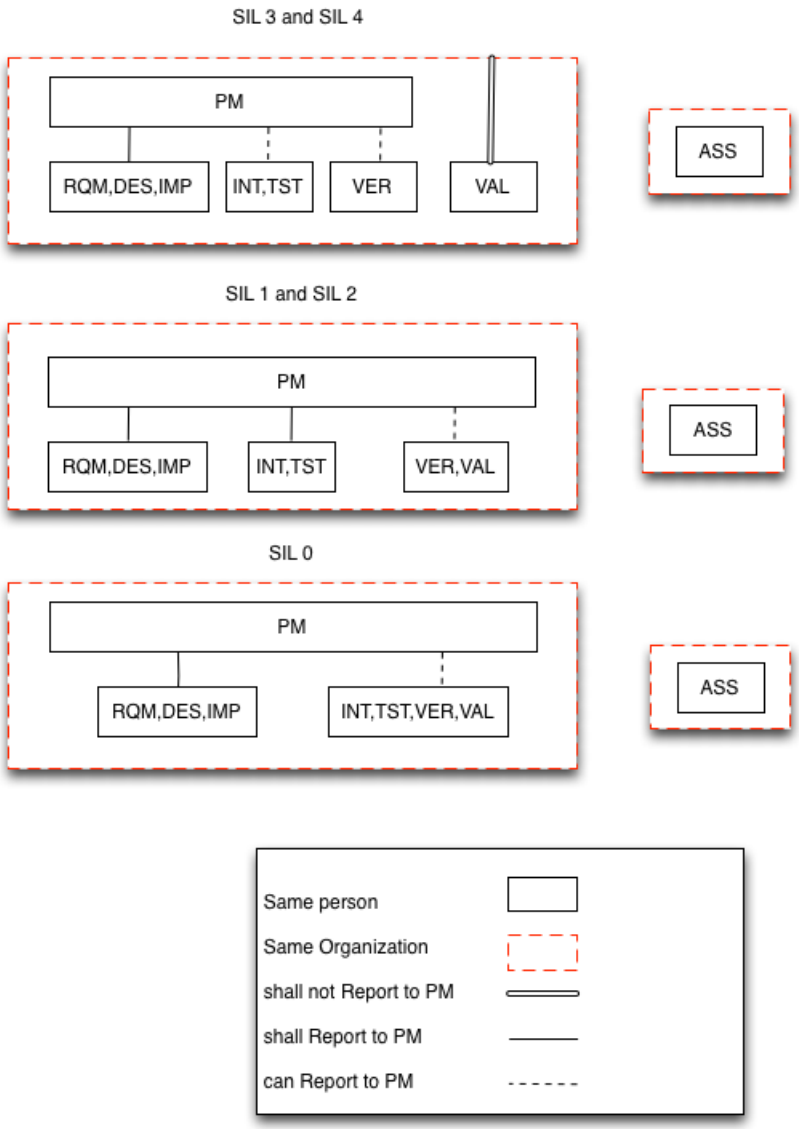


Figure 2.6: Organizational structure of the team members

- **Requirement Manager** (**RQM**), shall take care of the requirements management that analyze, trace and agree requirements.
- **Designer** (**DES**), shall creates specification of software artifact intended to accomplish goals
- **Implementer (IMP)**, shall implement the project
- **Integrator (INT)**, shall integrate the developing product inside the environment
- **Tester (TST)**, shall test the developing product

- **Verifier (VER)**, shall take care of the Verification of the Product
- **Validator (VAL)**, shall refers to Validation Activities
- **Assessors (ASR)**, shall be appointed by the supplier, the customers or the safety Authority in fact he shall be independent from the suppliers and chosen by the Safety Authority. The main purpose of the Assessors is to give agreement or disagreement for the software release in order to maintain the desirable quality level.

In SIL 4 Requirements Manager, Designer and Implementer for a software component can be the same person, he/she/they shall report to the Project Manager.

Integrator and Tester can be also the same person, he/she/they shall report to the Project Manager or to Validator.

Verifier shall be a different person form all the other rules and he can refer to both Project Manager and Validator. Moreover, in SIL 4 the Validator is not allowed to refer to the Project Manager and he has no influence on the validation decision but the validator informs the project Manager about its decision. Validator shall refer to an upper level, typically to management.

Safety Authority is a different company that guarantees and verifies the quality of a software development process. In order to establish a high quality Software, Project Manager and Validator shall not refers to the validation reports. The first one wants make process fast and the second one wants to make sure that the product is completely verified. Assessor and the company that is developing the product shall have not common interests.

Quality assurance procedures shall run in parallel with lifecycle activities and use the same terminology. The Software Quality Assurance Plan, Software verification Plan and Software Configuration Management Plan shall be drawn up at the start of the project and maintained throughout the software development life cycle [10].

2.2.6 Code Implementation

An Implementer has the aim of taking care of Software Source Code that shall be written following the basis of Software Component Design Specification. The code, basically, must be readable, understandable, and testable. It has to be written according to the requirement Specification. The use of Dynamic Objects and variable are not recommended and it's better to avoid their use. Also the use of pointers, recursions and jump can cause problems so they are not recommended. The complexity of functions, methods and subroutines is highly recommended to be maintained with limits and distributed [10].

2.3 Model-Driven Architecture

A **model** is a representation of one or more concepts that may be realized in the physical world, generally described within a domain of interest. Models can be abstract, mathematical and logical representations, as well as more concrete physical prototypes.

A **method** is a set of activities, techniques and conventions that implement one or more process.

2.3.1 MBE, MDE, MBT, MDT

Model-based Engineering (MBE) has been standard practice in many environments for many years. It is formalized application of modeling to support software life-cycle (requirement, design, analysis, verification and validation). The aim is to facilitate the process of system engineering traditionally based on a document approach. MBE improves the communication, the specification and design quality, the reuse of specification and design artifact. The output is supposed to be a system model where the emphasis is placed on evolving and refining the model using model-based method and tools [43].

Model-Driven Engineering (MDE) is based on the idea of representing systems using a graphical model. This approach has relevant benefit such as to understand and see the

system as a model and have the architecture of the system clear before starting deploying but the programming process remains still required, so efficiencies and cost saving are never materialized because of the complexity. The model in the Model-driven is completely sync with the process and connect. In fact, engineers constantly make custom programming changes directly into the system that improves also the model specification [36].

Model Based Testing (MBT) is a testing strategy used for model-based system. Testing consists in executing the System Under Test (SUT) for some particular inputs and in assessing whether or not the execution conforms with some requirements. The testing technique shall define the test cases to be submitted to the SUT and associate them to a decision procedure called oracle.

The oracle allows the tester to compute verdicts according to what the execution of the System under test reveals. The correctness is improved as much the requirements are respected. Test cases are a sequence of stimuli of the SUT and the oracle is based on a conformance relation[1]. When the set of Trace respects the requirements of the model the SUT conforms with the model. Model based testing is applied at different levels during the software development life cycle: Software integration testing,

High-level Code Verification and Object Code Verification. Software integration is performed by software technology on host computers. The software components can be easily simulated and tested. The tests for software may be generated but it shall be considered for each single function. Another application of Model Based Testing is based of verifying high level-code. Since the translation of the model into code is completely verified and certificated by the code generator it is not necessary to verify the outcomes of each transformation. [10].

The same model used for the translation can be used just verifying the consistency between the code and the model. The model-based testing approach can be used to create test cases conforming with the highest critical level of applicable CENELEC standards,

[1] A conformal relation is a mathematical relation between the set of execution on the SUT called trace .

in order to justify that the generated code is compliant with its model. Furthermore, the generated results may be formally verified against the model [43].

Model Driven Testing (MDT) derives the tests that the system implementation will need to make sure it behaves as expected, starting from models that are used to specify the system. When the software is derived from the model, there is no need to test the code unless our-code generator is incomplete or unreliable. Instead, we should test the models themselves. In that sense, before developing a model, it is necessary to write the model test that will be used to evaluate the new functionality at the modeling level.

Test-driven development (TDD) relies on the test-first philosophy. The Developer first writes executable test cases that will be used to check the new functionalities to be implemented and then if the test fails, he writes the code needed to pass the test. As soon as the new code passes the full test suite, the code can be refactored and the process starts again [35].

2.3.2 Model Driven Architecture Standard

The Model Driven Architecture (MDA) as it is defined by the Object Management Group (OMG) provides an approach for deriving value from models and architecture in support of the full life cycle of organization. Moreover, it represents and supports every step from requirements to the technology implementations. By using MDA models, the complexity of large systems and interaction is smoothed.

MDA uses models to enhance the agility of planning, design and to improve the quality and maintainability of the resulting products. It can also be used to relate the models to implements, solutions and related artifacts.

Int the OMG's MDA standard there are four principles to follow:

- Model shall be expressed in a well-defined notation to improve the understanding of solutions

- A system can be organized around a set of models by imposing a series of transformations between models, organized into an architectural framework of layers and transformations.
- A formal description of models in a set of meta-models is required to improve the meaningful integrations and transformations
- Acceptance of this model-based approach requires industry standards to provide openness to consumers.

The MDA set of standards includes the representation and exchange of models in a variety of modeling languages, the transformation of models, the production of stakeholder documentation, and the execution of models. MDA "connects the dots" between these different viewpoints and abstractions.

System is one of the key concepts of modeling in the Model-Driven approach. Developers must distinguish it from technologies like software source codes. A Model is an excellent way to represent, understand and specify systems.

A **Model** is a collection of information representing some aspect of a system based on a specific set of concerns. The model is related to the system with an explicit mapping. The relationships between these models provides a web of dependencies that record the process by which a solution is created and help to understand the implications of changes at any point in the process.

Models, System, modeling and Model Driven Architecture (MDA) are the basis for a set of development approaches known as model-drive development (MDD). MDA is largely discussed and improves the quality, efficiency of the software development [18].

System development can be organized around a set of models by imposing a series of transformations between them, organized onto an architectural framework of layers and transformations. To support this approach the OMG has defined a specific set of layers and transformations that provide a conceptual framework and a vocabulary for MDA.

OMG Standard identify four layers, they are shown in figure 2.7:

- **Computation Independent Model (CIM)** is a target model containing all the business rules defined in the core business model.

Platform Independent Model (PIM) A model independent of the specific technological platform used to implement it

- **Platform Specific Model (PSM)** A model linked to a specific technological platform, indispensable for the actual implementation of a system.
- **Implementation Specific Model (ISM)** An actual Implementation model specific for the platform, directly translatable.

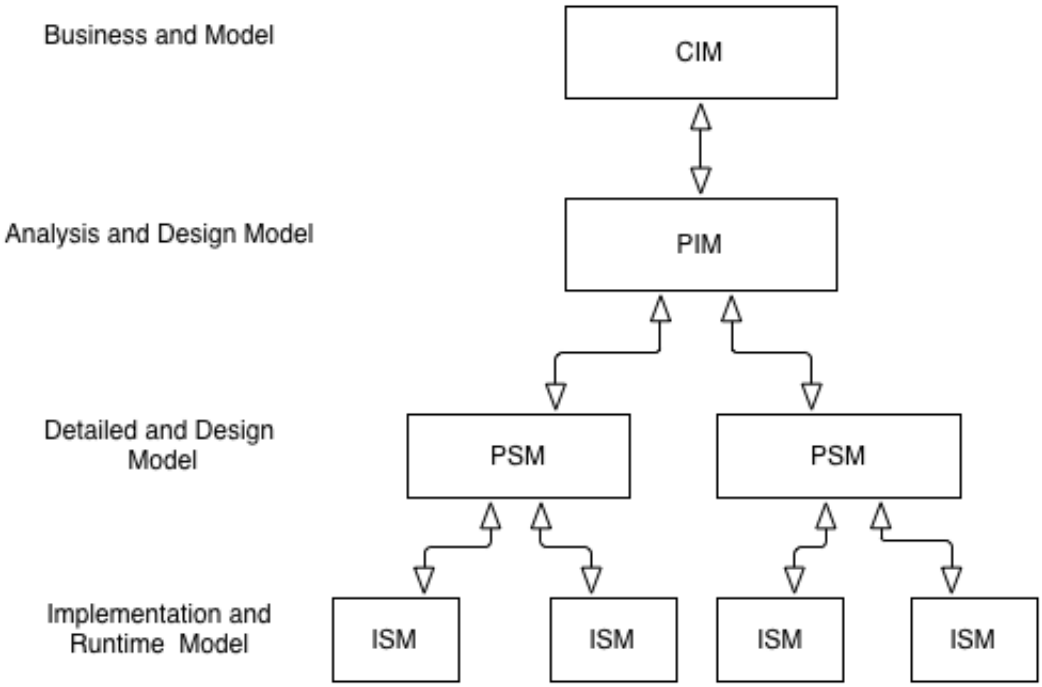


Figure 2.7: The layers and transofrmations of MDA

A key aspect of the MDA method is to recognize these transformations applying them to abstract descriptions of a system, in order to obtain development as a series of refinements. MDA, generally helps developers to understand what is essential to the solution being designed separate from the detail of the “platform”.

The transformations between models become first-class elements of the process because a lot of work is required to define transformations. Models are so described as a set of meta-models. The ability to analyze, automate and transform models requires a clear, unambiguous way to describe semantics and notations.

The OMG consider the importance of meta-models and formal semantics essential for their practical use. For this reason they set up a **Meta-Object Facility (MOF)** that formally define the abstract syntax of a set of modeling constructs[18].

The MOF provides a metadata management framework, and a set of metadata services to enable the development and interoperability of model and metadata driven systems. This MOF specification integrates and reuses the complementary UML 2.0 infrastructure submission to provide a more consistent modeling for Model Driven Architecture. UML 2.0 provides the modeling framework and notation [27].

MDA places modeling at the heart of the software development process. Various models are used to capture various aspects of the system in a platform independent manner. Sets of transformations are then applied to these PIMs to derivePSMs. These PSMs need to be eventually transformed into software artifacts such as code, deployment specifications, reports, documents, etc. The MOF Model to Text (mof2text) standard addresses how to translate a model to various text artifacts.

An intuitive way to address this requirement is a template based approach wherein the text to be generated from models is specified as a set of text templates that are parameterized with model elements. A template-based approach is used wherein a Template specifies a text template with placeholders for data to be extracted from models. These placeholders are essentially expressions specified over meta-model entities with queries being the primary mechanisms for selecting and extracting the values from models [18].

2.3.3 Modeling Languages: UML and SysML

To be useful, any model needs to be expressed in a way that communicates information about a system among involved stakeholders that can be correctly interpreted by the stakeholders and supporting technologies. The structure, terms, notations, syntax, semantics, and integrity rules that are used to express a model constitute the modeling language. Well-known modeling languages include UML, SysML, SQL Schema, BPMN, E/R, OWL, and XML Schema [15].

UML

By means of abstraction, Unified Modeling Language (UML) is a software language that allows complexity reduction. Using, in fact, an appropriate level of details, UML gives facilities assessment of the key characteristics of the design on the basis of representation. It has been standardized as a modeling language. The Models are written in terms of one or more diagrams types, classified as structure diagrams and behavior diagrams.

SysML

The Object Management Group's OMG SysML is a general-purpose graphical modeling language for representing systems that may include combinations of hardware, software etc. The language is intended to specify and build systems using UML (for software design). SysML can represent a lot of system specification such as: Structural Composition, state-based behavior, constraint, allocation between behavior, Requirements and their relationship, test cases, design elements etc. It is derived from UML, which was originally specified as a modeling language for software design, extended to support general-purpose system modeling. About half of the UML language specification was reused [15].

The SysML specification defines a set of language concepts that can be used to model systems. They can be classified into three parts:**Abstract Syntax, Concrete Syntax** and

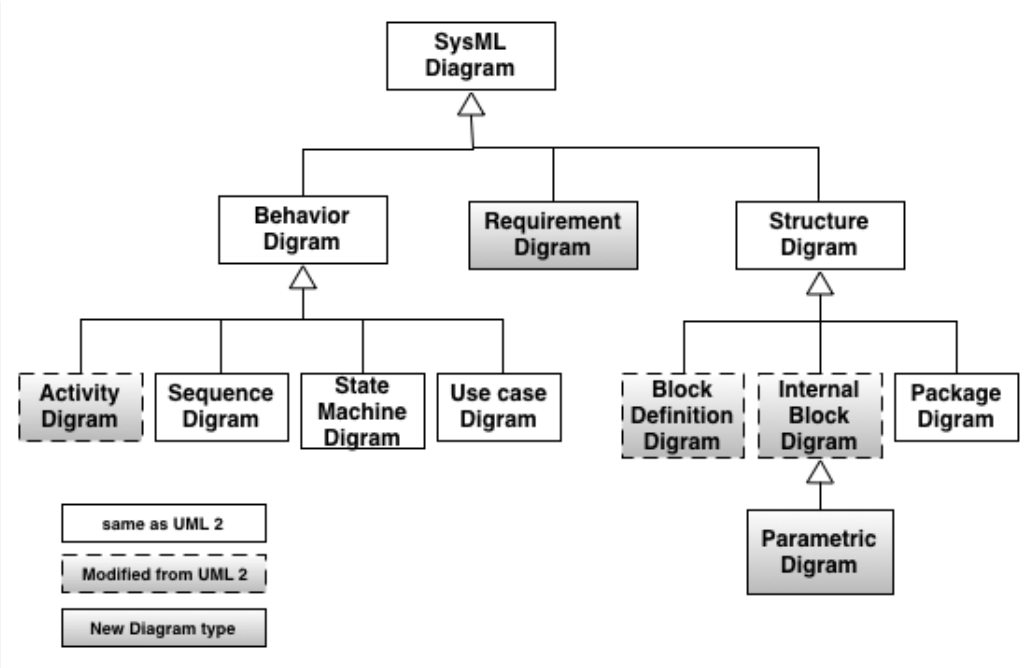


Figure 2.8: SysML Diagram Overview

Semantic.

Figure 2.8 shows different types of SysML diagrams with their organization.

- **Package Diagram:** organizes models in term of packages
- **Requirement diagram**: represents text-based requirements and their relationship
- **Activity Diagram:** represents behavior in terms of the order of actions
- **Sequence Diagram:** represents behavior in terms of a sequence of messages exchanged between systems
- **State Machine Diagram:** represents behavior of an entity on terms of its transitions between states triggered by events
- **Use case diagram:** represents functionality in terms of how a system is used by external entities
- **Block Definition Diagram:** represents structural elements called blocks, and their composition and classification (UML Class Diagram). It is composed of **Blocks**,

that describe composition relationships between blocks. It identifies the usage of its type in a context.

- **Internal Block Diagram:** represents interconnection and interfaces between the parts of the block (UML composite structure diagram) It is composed of **Parts** which is the fundamental modular unit for describing system structure in SysML: It can define a type of a logical or conceptual entity. Parts is a type that is described also as a set of similar instances, or objects.

- **Parametric Diagram:** represents constraints on property values

SysML has its own graphical modeling languages that fits for system engineering and is increasingly used in industries. The standard is not always precise enough and leaves an important part of its semantics undefined. For this reason, Model Driven Engineering shall be adapted to the SysML language in order to suppress any ambiguity in the system models and to allow formal proof. This requires the addition of new modeling constructs and the definition of a set of rules to clarify some semantics variation points of SysML [40].

2.4 Tools

CENELEC Standard EN50128 provides three classes of tool respectively called T1, T2 and T3. Tools Class T1 are all the tools present in the process that generate no outputs which can directly or indirectly contribute to the executable code and data of the Software. Tools Class T2 are all the tools present into the process that support the test or verification of the design or executable code, where errors in the tool can fail to reveal defects but cannot directly create errors in the executable software. Tools Class T3 are all the tools present into the process that generates outputs which can directly or indirectly contribute to the executable code and data of the safety related systems [10].

A large number of different tools from various backgrounds are available which mostly handle on means of description and allow applying one or more methods for different steps in the development process. Since the functionalities tools support can be used in different methods and for different development steps the use of a tool can be necessary at different times during the product lifecycle. Moreover, some tools are able to use combined different methods. The focus of this comparison is on the methods of the code translation [10].

A lot of companies, in fact, use commercial tools like Artisan Studio, Atelier B, Rational Architect, SCADE or Simulink, to build their software models by using the implemented means of description and the implemented methods and to generate reliable source code in C language. There are, also, a large number of existing open source tools which usually have an academic background, Therefore most of them are not certified, but as some of these tools have been around for quite some time, they could be suitable to fulfill all requirements. For this reason, it is possible to use open source software combined with support through an external partner that can provide the needed level of customization and service support required by companies.

One of the main aspects of developing a well-defined process is to take care about the inter-operation between tools. In this process, in fact, all the used tools shall share some information. Starting from a consistent traceability every tool shall preserve and forward it to the next in the tool chain. Every tool inside tool-chain, since they are not certified, cannot directly contribute to the executable code so they are classified into Class T1.

2.4.1 Papyrus-SysML: semi-formal modeling tool

Papyrus-SysML is a general-purpose graphical modeling language based on UML 2 that support analysis, specification, design, verification and validation of systems that include hardware, software, data, procedures and facilities.

SysML uses Structural composition, interconnection and classification, functional, mes-

sage and state based behavioral, allocations between, structures and constrains, requirements and their relationship to other requirements. It is based on the OMG specification[36], which describes a visual modeling language that provides both Semantics and Notation but it does not describe neither a tool nor a method. SysML is based on Blocks, which represent functions or concepts m, they are related with links and composition, internally arranged or acting. The aim is to describe a system made out of sub-system compositions and also it surrounding, requirements and behavioral.

The scope of the Project using SysML is to manage the gap between prose system analysis and formal model dedicated to the design of the OBU. The team shall provide a high level model associated to the system analysis, SSRS and API. A Block Definition Diagram and a Package Diagram, describe physical and functional architecture. Data definition will be defined with Block Definition Diagrams.

2.4.2 SCADE: a modeling tool with a SIL 4 code generator

SCADE system[2] and SCADE suite[3] can be used as default approach to develop the formal model on the on board Unit. The Sub-System Requirement Specification is a document, which shall define the scope and the structure of the sub-system and manage the requirement allocation on sub-systems and functions. SSRS shall define the interfaces of the systems and sub-system, external and software interfaces according with the API provided and hardware specified. It also shall cover all the requirements and the function of the input documents, clarifying the SRS SUBSET-026 and take into count the design of the OBU.

SCADE System

Scade System is a system design and modeling tool for embedded software code production. It is based on the use in critical systems with high dependable requirements and

[2]http://www.esterel-technologies.com/products/scade-system/
[3]http://www.esterel-technologies.com/products/scade-suite/

provides full support of industrial systems engineering processes such as EN50128. Since it is a platform for deploying model-based, base on SysML architecture and provide easy -to-use editor to decompose requirements it is the best choice as a bridge from SysML model on Eclipse and Scade Suite for model-based programming. It provides also some restriction to the SysML OMG but also it extends some rules in order to improve the language and to have less ambiguity.

SCADE Suite

Scade Suite is a model-based development environment. SCADE Suite is the integrated design environment for critical applications spanning model-based design, simulation, verification, certified code generation, and interoperability with other development tools and platforms, including requirements traceability. Scade is based on CENELEC EN50128 certified at SIL 3-4 for the Rail Transportation. Scade Suite offers a Graphical User Interface where engineers can model their system and program it starting from the models. This software provides also a simulation tool to check the behavioral with a graphical interface and allows the user to translate the code in C language according with critical and safety system based on standard EN50128.

2.4.3 Verification and Validation Tools

As the goals of the project include the selection, adaption and construction of methods and tools for a FLOSS[4] development in addition to performing actual development steps, differing from the plan for a full development project, the plan covers also activities evaluating the suitability of methods and tools, and it makes previsions for incorporation of V&V of partial developments which are actually done. V&V shall be done in a FLOSS style, and it has to suit a model-based development.A further main consideration shall be done to respect the requirements of CENELEC EN50128.

[4]Free and open-source software

On one hand it designs a tool-supported methods with which the software can be developed and maintained. On the other hand it performs a part of the development or representative parts of the design. In the Verification and Validation steps there are a lot of aspects to take care on (for example the functionalities and the architectures of the system and sub-systems). But, also, the external and internal interfaces of the sub-systems, the software components, the performance and Safety objectives and constraints, functional and safety properties.

Verification is the activity to ascertain that a particular step in the development has achieved its goals and its result correctly refines or implements its input, which may be a higher-level design or a specification. The objective of this is to examine and arrive at a judgment based on evidence that output items (process, documentation, software) of a specific development phase fulfills the requirements and plans completeness, correctness and consistency.

MaTeLo is a test case generator tool for systems. A probabilistic model describes his behavioral. It is based on its own test models called "usage models", which are made out of transitions that represent the stimulus of the system, and the control applied to the output obtaining expected results. Since MaTeLo uses the **Markov Chains** theory designed to reduce the costs and time of test phase, transitions are separated each other. The states are stable indicating the state of the system before the stimulus. MaTeLo allows the process to submits consistent test cases to the SUT. Moreover MaTeLo has three main functions: test modeler, test case generator and test campaign analyzer.

Usage Model allows the implementation of test models that describe the use cases of the SUT completed with the point of view of the tester . The test cases now can be generated automatically. It contains several test generation algorithms that can be used for different purposes. In the OpenETCS project the System under test model is an on-board EVC based on the SSRS Subset 026. The final goal of the Verification and Validation team using MaTeLo is to take into account all the possible signals, rules and pass through all possible states. A test model can be considered as a development artifact in the same

way as a system model. Since MaTeLo model is built from system specifications, some ambiguous points in the test can occur.

Once the MaTeLo test model is performed and testing strategy is defined it generates test cases. This tool contains also different test generation algorithms with really different scopes. Probability of each model transition refers to the possible number of occurrences of those stimuli [6].

Test Generation Strategy

Most Probable Represent the Matelo test which is the most probable. It generates a set of scenario with the highest probability of occurrence. It has a simple complexity even because it is deterministic and cover just nominal cases [2].

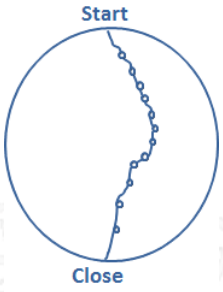


Figure 2.9: Most Probable

Focus approach A test profile brings high probabilities to the system focuses part. It generates test cases focused on these parts. It could be simple and/or complex with a random generation approach [2].

Figure 2.10: Focus Approach

Arc Coverage This algorithm provides an optimum test suit which covers minimal numbers of test steps that covers both transitions and requirement. It is complex and almost statistic but based on requirements [2].

Figure 2.11: Arc Covarage

Operation Coverage A test profile reproduces the environment profile. Matelo generates test cases close to the customer behavior , It take into count interactions and is basically random [2].

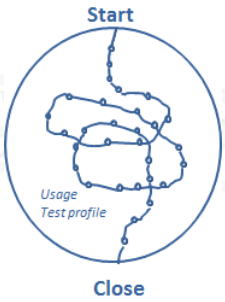


Figure 2.12: Operatio Coverage

Diversity and RT-Tester

Diversity is a model-based testing tool developed by CEA. It identifies for each possible execution of the program the constraints to be satisfied. It takes as input a model of complex systems corresponding to specification level and lower level design. Models are described with the help of state-flow type language and communicating automa. **Diversity** can be seen as a white box testing tool. The first step to consider is to define a requirement model in the form of a sequence diagram or a MaTeLo test scenario. A

symbolic tree shall be created where each path detonates a possible execution of the sequence diagram. These paths could be ideally infinite considering the loops but Diversity has some stopping methods. Since it is really complicated to test entire systems, there are some facilities that allow users to test subsystems extracting symbolic tree. Since a fault of the entire system can be a fault of at least one of its sub-systems, so the conformance of each sub-system guarantees the conformance of the whole system. These two tools, MaTeLo and Diversity, are used in cooperation in order to reduce the test cases. MaTeLo in-fact analyze the model as a black box and generate the test cases, Diversity instead analyze the formal model as a white box using the semi-formal SysML model in order to filter the testes generated by the first one. These two tools create different kinds of models. Such models are useful to describe executable behaviors very close to the actual implementation. Moreover it is mandatory a traceability between the tests in order to respect the safety requirements imposed in the process[6].

RT-Tester is a tool created by Verified that creates and performs test generation and makes a real time evaluation. It support different testing approach such as unit testing, software integration testing for components and automated test case generation, automated test data generation, procedure generation, Requirement Tracing. RT-Tester generates from the semi-formal specification in UML-SysML test cases and submits them to the System Under Test. The test procedure is made in combination of test oracles. Reach statement coverage and branch coverage are generated automatically. Since it uses the SysML model all the diagram are completely linked to the requirement. RT-Tester produce documentation based on two main documents: Test procedure specifies how a test case can be executed and test reports that expresses all the important information [6].

2.4.4 Git-Hub and Revision Control

One of the main aspect of software development projects is to take under control the versions of the products that they are developing. Companies adopt Version Control Software to take care of the Revision on the product and to understand the progress of the project

at the moment. The easiest way to have a Version Control is to create a directory for each version in the Local File System but this approach is incredibly error prone. **Centralized Version Control System** (CVCS), uses a shared file system, even if everyone of the project knows the current version, it has a lot of downsides such as the single point of failure and the concurrency work is not allowed. **Distributed Version Control Systems** (DVCS) resolves some problem of CVCS with fully backup.

Furthermore, DVCS deal with having remote repositories where Developer can work with and collaborate with different groups of people in different ways simultaneously within the project. Git is one of the most famous **Version Control System** and one of the most used in the Software Development. It has a lot of new features that helps the Developing Team to take care of the Version Control of the Software, the issues and the team members. **GitHub**[5] is the distributed version of **Git**.

It is really useful for very distributed team with different skills, in fact it offers a lot of features to communicate as much as social networks like feeds, follower and very readable graph. The Major difference between Git and any other VCS is the way Git thinks about its data. Usually the VCSs ,every time there is a commit, make a snapshot using the unchanged file as reference with just a link to the previous one. Indeed, Git, such as a file-system copies for each version the new files. Every thing in Git is checked-summed before it is stored to prevent corruptions with SHA-1 hash. Git has three main states in order to maintain the data: committed, modified and staged. This leads to the three main section of a Git Project: Git directory, working directory, and the staging area.

2.5 Agile-Method: Scrum

Scrum represent one of the more used agile methods by companies and it is still in expansion (Figure 2.13). It is iterative, indeed it is based on development cycles called Sprints. These Sprints have a fixed length in terms of time and they take place one ofter the other

[5]https://github.com

even if the planned work is not completed. The matter of each sprint has to be clear before it starts, and every certain period of time the team has to describe what is done and what is planned to do [41].

At the end of the Sprints, a review of all the work done in this slot shall be done and the team receive feedback. The traditional way to build software is the "Waterfall" method but it has different lacks and typically it is not flexible enough to support some process. This approach, once stakeholders have reviewed and approved the plan the team start working and future changes of the requirement are not allow.

Moreover, all the ideas come at and all the phase are strictly defined and one come after the other. Scrum is really based on the people involved in the development since it is an evolution of the iterative and incremental Systems Development Life Cycle. One of the main aspects of Scrum is the Product Backlog which is a prioritized list of what is required to do. It evolves over the lifetime of the project and items can be added and removed or repositioned [39].

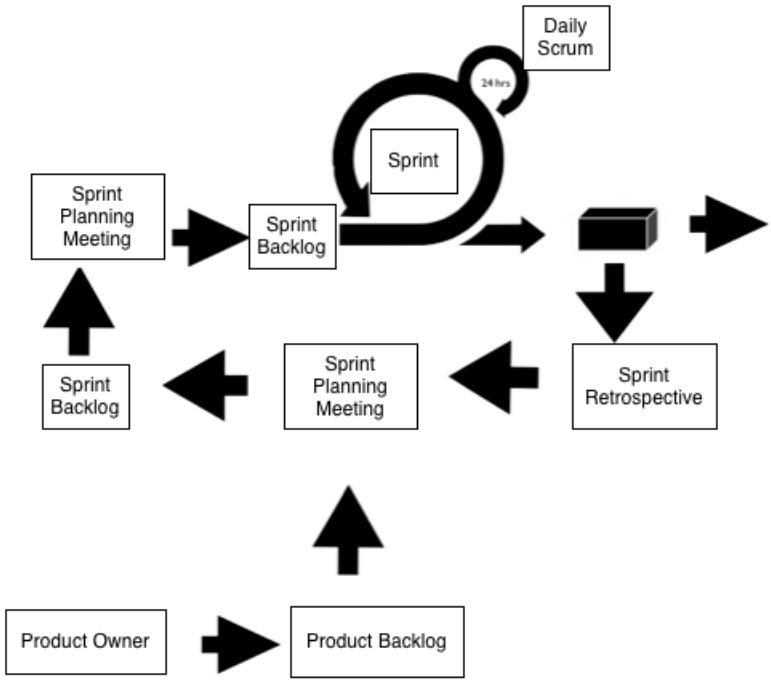


Figure 2.13: Scrum Scheme

2.5.1 Scrum Team Members

There are three main role in this Agile Method. These are The Product Owner, The Team and the Scrum Master.

The **Product Owner** is the responsible of the project and that this development works fine. He is the master of the prioritized list called Backlog so he can decide the priority of the next works and the topics for the next Sprints. The Team is composed of a worker that actually build the product. They have a lot of autonomy, accountability and include all the necessary expertise. The team is composed of at most 9 people and it include for software development programmers, engineers and tester. Every team should be focused just on one goal of the project [34].

Scrum Master is the one who serves educating and guiding the team in the use of Scrum and protecting them form outside interfaces. The Scrum Master and Product Owner cannot be the same person since the first one has to decide if the pending requests from the Product Owner can be accepted or not. The Product Owner may delegate the work to Products Owners on subordinate teams, but all decisions and directions come form the top-level Product Owner [34].

The **Product Backlog** is the entire point of view of everything that must be done in order of priority. This evolves with the time running and gives a plan for the teams. The product owner is the one who has to care about the sorting of the next works of the team. He/she, continuously, updates the backlog with new steps and needs of the team and the Scrum Master.

To order the priority there are some rules in the backlog based on the speediness of the team, on maximizing the return of investment and the business value. The product Backlog has to specify works scheduled but the detail of how the work shall be done are not required. More details can be added but it is usually not recommended [34].

Every **Sprint** starts with a meeting called Sprint Planning Meeting. It is divided into Sprint Planning Part One and Sprint Planning Part two. Inside the first par the Product

Owner and the team selects the high-priority items and discuss their goals. "Definition of Done" shall be very clever to every component: Done means coded to the standards, reviewed, implemented with unit test-driven development, tested with 100 percent test automation, integrated and documented. The second part Sprint Planning Part Two focuses on how to plan and implements the items decided to work on. At this point all the actual schedule of the work shall be take into consideration.

The Product Owner can also decide to schedule before lowest priority task than other if he or she consider them easy to complete. The Product Owner shall also to estimate the time slot in which the work will be done, if the team has experience on that and/or if them know the technology they are working on.

At the end of the meeting a schedule must be planned and the estimation time to complete them too. The list is the starting point, but more tasks will emerge as the Team work on each task of the Product Backlog. The Team chooses the ordering of Sprint Backlog tasks to maximize the velocity of production and quality of the work. Scrum adopt the rule "go to where the work is" in fact it prefers multi skilled worker rather then heavy expert in just one field in order to spread the experiences all over the Team and everyone can learn form the others [34].

Work plan

Every day, the Team members update their estimate time remaining to complete their current task in the **Spring Backlog** and drown a graph that explain how much time remain until the work is complete. This graph should express also the working time until that specific moment in order to express how much work remains to do. This graph is usually a sloping graph that is on a trajectory to reach "zero effort remaining". A **Scrum emergency Procedure** is planned when in the mid-Sprints the line is not going downwards [39].

This procedure provide:

- Change the approach to the work and remove impediments
- 2- Get help by having someone outside the team
- 3- Reduce the scope of the work
- 4- Abort the Sprint

Grooming is a meeting which the Product Owner has to care about to order the Sprints and it takes between 5%-10% of the time to detailed the requirements analysis, splitting large tasks into smaller ones etc.

After the Sprint ends, the Sprint Review is scheduled. In this meeting the Team gives and receives some feedback and this is a good moment for the Product Owner to understand the skills of the team and the state of the art.

It is an opportunity for the Team member to discuss what is working and what is not working. Scrum moves groups away from the older project-centric model toward a continuous application development model. The old understanding of the software development is not still active in Scrum, in fact the **Project Manager** is not present and the project will not have a traditional evolve.

Scrum is a framework that provides a lot of degree of freedom to the Team and it is an important mechanism to understand what are the impediments that are impacting the development. The Scrum will quickly reveal the weakness of the project and provide a way to solve it in short cycles [34].

Capitolo 3

OpenETCS

"The purpose of the OpenETCS project is to develop an integrated modeling, development, validation and testing framework for leveraging the cost-efficient and reliable implementation of European Train Control System (ETCS). The framework will provide a holistic tool chain across the whole development process of ETCS software. The tool chain will support the formal specification and verification of the ETCS system requirements, the automatic and ETCS compliant code generation and validation, and the model-based test case generation and execution." [3]

3.1 OpenETCS Methods

Standard **CENELEC EN50128** lists a number of methods (fig. figure 3.1), process and tools to use in the development for railway Software. Starting from Software Requirements Specification activity, Engineers shall analyze Requirements first in natural language and then translate them into a formal or semi-formal model.[1] During all the development process, this formal description shall be translated back into a natural language to provide all kinds of needed documentation.

The OpenETCS process follow seven general functionalities:

[1] A formal approach is a way to describe system or software that builds upon rigorous syntax and rigorous semantics. (Like mathematics formulas)

- Transformation of textual specification into formal specification
- Transformation of formal requirement specifications into formal software and software module design and architecture descriptions.
- Source code generation
- Verification of the models and source code
- Validation
- Creation of documentation
- Terminology management / Intelligent Glossary

In order to develop a software based on safety-critical railway rules, the OpenETCS process must give a formal approach to the requirements creating Models, as it is Highly Recommended in the Standard.

The **syntax** describes how the system or software description is built and valid. It is usually made through a grammar and a set of constrains. It can be textual or graphical.

A **semantic** gives a meaning to each object on the system or software description. A mathematical model is given as semantic to rules and defines how those objects interact each other. All the models have a very different formal approach. An explicit document should be provided to describe the semantic that can be informal or formal.

The use of **formal methods** is because of the expectation that performing appropriate mathematical analysis can improve the reliability and robustness of a design. Compiling or running a language is not enough to give it some semantics, as this semantic is hidden within the execution/compilation step. [2]

Also, **Semi-formal** approaches [3] have a precisely syntax but semantics is not precisely defined, It can become formal rigorously defined through a mathematical model.

[2]For example, ESTEREL SCADE uses Lustre which is a formally defined language with synchronous data flow programming for reactive systems.

[3]such as SysML/Matlab language

Formal approaches are specialized in verification of some kind of propriety and have some restriction on the type of the software. The usage of these can impose specific resources in every different phase and impact the entire system process. At the end the development process shall design the benefit introduced from the formal approaches to all the system process.

Formal approaches give some efforts to the process:

By using a non ambiguous notation, the designer shall clarify his mind and enhance the understanding of the system.

It is able to verify some properties in an exhaustive way.

By verifying properties along the construction cycle of a system it is easy so ensure that some formalized requirements are fulfilled in the final software.

The formal approach can be developed into four main stages: Formalization of Requirements, Design support, Implementation and Verification.

By enforcing a non-ambiguous meaning for all parties and all the members of the team enrolled in the development a formal approach is applied to the **Software Requirement Phase**. In the **Design and Architecture Phase** a formal approach shall support the system design and architecture design. **Verification and Validation** can be also supported. Depending on the method the actual system source code can be generated by a formal model, derived or written manually.

A Software is generally divided into a set of operations: procedure, methods and function. To each of them a precondition and a post condition are associated, and some conditions should be guaranteed. With this approach if all possible execution guarantees the precondition and post condition the operation is considered correct,

The model checking of concurrent and synchronous languages is an approach model-based, related to the state based model and data flow. It uses a temporal logic, which is usually a first order logic that expresses the relationship between events.

Static analysis of software code is based on the idea of transforming the domain of con-

crete program variables into a simpler domain. Then the analysis is done for all possible paths within the abstract domain using a specific mathematical approach.

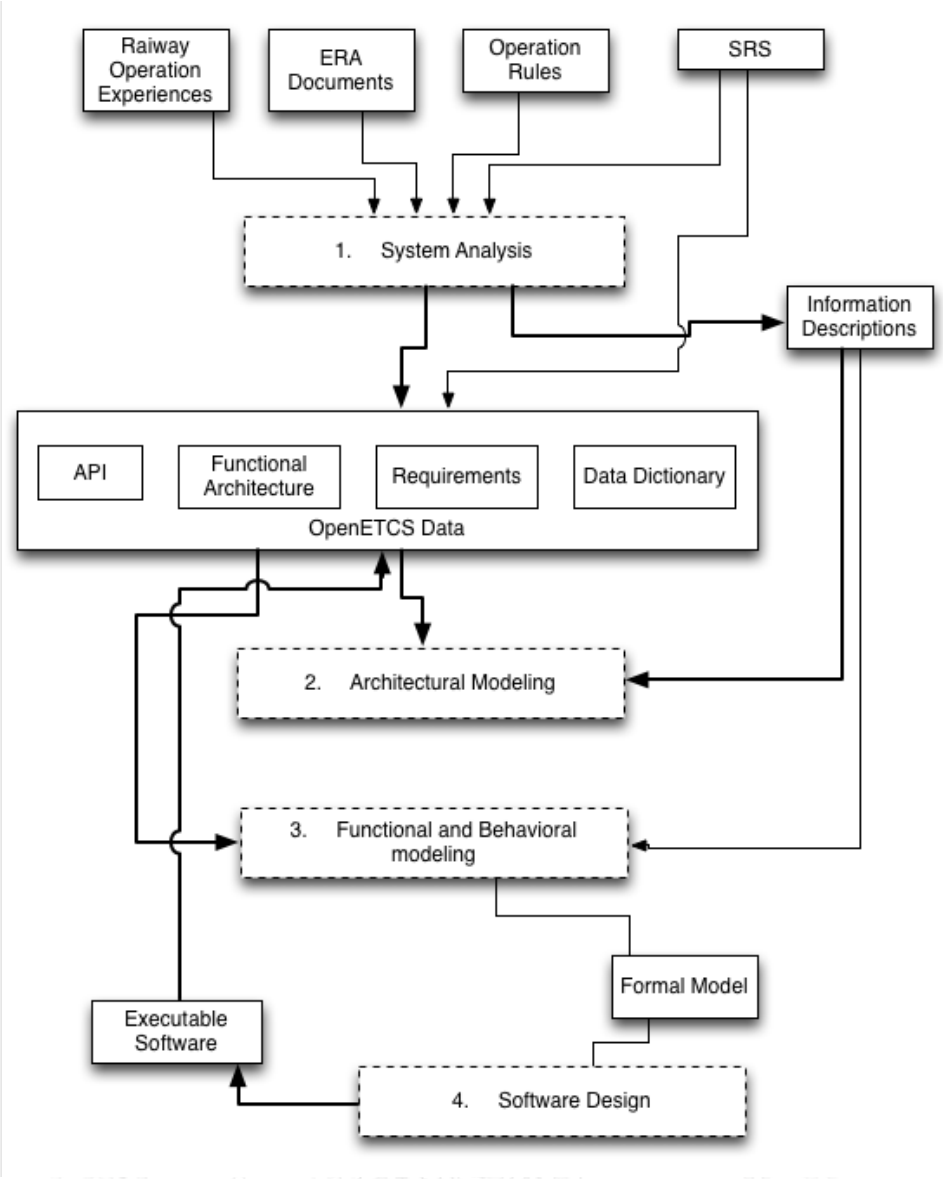


Figure 3.1: OpenETCS project approach

3.2 OpenETCS Process

Using the **UNSIG SUBSET-26** OpenETCS shall give a semi-formal specification of the ETCS on-board Unit to enhance the understanding of the subset. Moreover, OpenETCS shall be able to animate the model for testing and analyzing at system level, to provide

information on the completeness of analysis of the SUBSET_026.

The **OpenETCS process** is strictly based on validation of safety design. In order to minimize the number of different models, which can be reused by railway users at the end of the project, the process shall take into account the safety concepts from the first step. The most important elements of the System life-cycle of EN50129 and the Software development life-cycle of EN50128 are the separation of the software development life-cycle into well defined phases and producing a lot of documentation related. In order to do that every participant shall have his own assigned part and all the life-cycle process quite clear.

OpenETCS process (fig. figure 3.2) shall analyze the input documents including the Sub-set 026 and pro-vide semi-formal models. At this moment the process is split into two sub-processes. On one hand there is the real safety process with all the software con-straint.

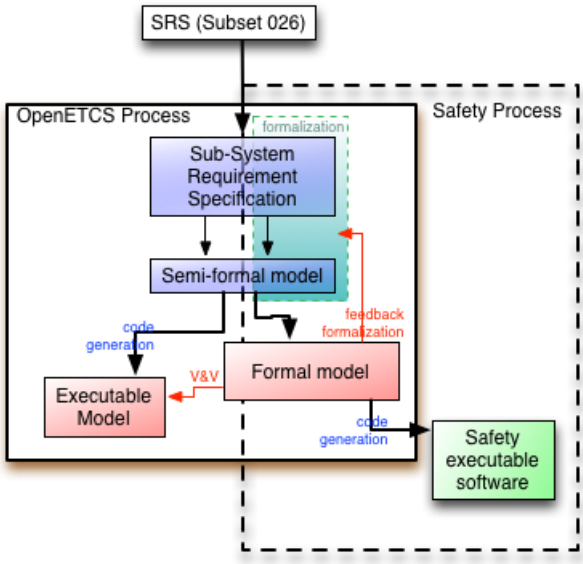


Figure 3.2: OpenETCS process

On the other hand there is the non-safety OpenETCS Process that does not include the formalization activity. This division is made to improve the speed of the process, giving very fast an executable model by using a code generator. This code, of course, is not safe

and standardized. A formalization activity is the main step of the **OpenETCS Safety Process** (fig.figure 3.3), which produce a Formal Model. This model gives some feedbacks to the formalization activity but also to the Verification and Validation approach. A code generator of the formal model produces Safety executable software.

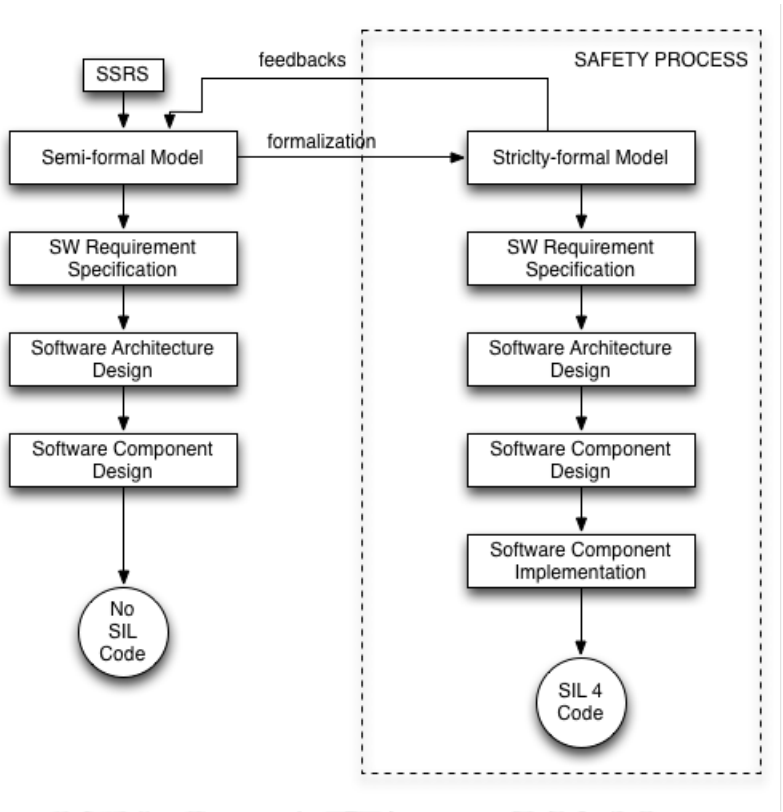


Figure 3.3: OpenETCS process

3.2.1 OpenETCS Requirement Analysis

The aim of this phase is to analyze the input documents, create and define a model of the subsystem. In order to do that a set of requirements which describe the functionalists of the sub-system shall be retrieved and a description of the interfaces provided by the APIs of the sub-system shall be done. According to the CENELEC standards, n SSRS that define the structure of the subsystem and an API document is enough to complete the input documents needed.

Requirement engineering is a hard technical process. A requirement shall be readable.

Giving a good structure of requirements can make a huge effort to management and traceability.

Since the openETCS process is based on the (SRS) ETCS Subset 026 [4] , it shall be analyzed, structured and well understandable for all the Developers of the project. This SRS covers 8 chapters, often the specification offers multiple solutions on how to implement a specific function. A requirement can be mandatory identified of the using of the word "shall" and optional referred the using the word "may". The ERTMS/ETCS on-board equipment shall implement all mandatory requirements.

Relationships between Requirements, or Requirements and model, allow system engineers to have a clear view of the system and models. A possible method is to use hyperlinked documents, where statements can be highlighted, linked to other statements in either direction. Linking statements together using some kind of database support achieve the simplest form of traceability. Linking information has to be held separately from the documents to be helpfull. So the statements are independently and uniquely identifiable.

In spite of this a database of requirement coming from SRS document are organized in files. This file has been created by means of the tool Doors.[5]

3.2.2 Modeling

The approach used by OpenETCS project is the document-based system engineer, which is characterized by the generation of textual specification and design documents. The Developers team decides, at least for the first stages, to use a combined use of **Model-Driven and Model-Based Architecture** depending on the need of the process and the tools.

The advantages of this decision are:

Real-time changes to business rules, forms, workflows, life-cycles, and the data model

[4] A document by the unified European Train Control System

[5] Doors is a mulch-platform, enterprise-wide requirements management tool designed to capture, link, trace analyze and manage a wide range of information.

without complex programming of model synchronization issues

Graphical solution development with dynamic schema

Separation of business logic and technology

Lower system Complexity and reductions on cost to deploy

First step of modeling (fig.figure 3.4) is to define the main function considering the behavioral of the model. According to the first proposal of the team the On Board Unit software design is divided into 73 main functions. Each one of this shall be integrated in the OBU software with every characteristic, interfaces and data type. With these functions the developers team shall divide, test and schedule the whole process, step by step. The safety process gives a mechanism to verify and validate the semi-formal modeling process by doing a safety Verification. All the results of the formalization activity must be analyzed.

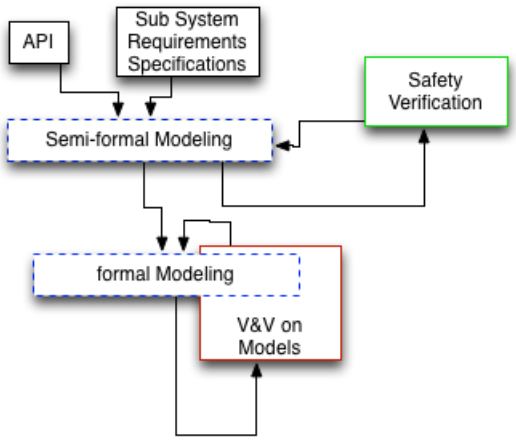


Figure 3.4: OpenETCS Modeling project step

Data Dictionary

All the Data types used in the project are preserved into a Shared repository called Data Dictionary. It is used as a reference for specification, modeling and V&V activities. It is possible to divide the data into two macro parts. Data coming from the "outside"[6] specified in ETCS Language and On Board Unit Internal data types.

[6]data received from the truck, train, Divers, National System, Radio etc.

Data Dictionary is organized, as it is shown in figure 3.5, with all the interaction between parts and shared information.

The ETCS Language types are defined in chapter 7/8 of the SRS SUBSET 026, where there are definitions named “Packets” and “Messages”. Messages are composition of Packets, moreover Packets are a composition of Primitive Types. These definitions reflect the data format and external interfaces (especially in the air gap between the track and train). Data formats should save bits during transmission making length very different from each other.

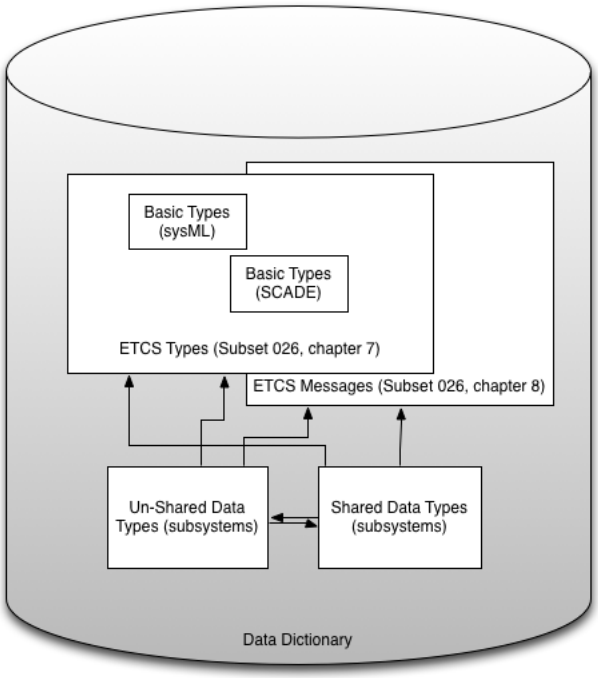


Figure 3.5: OpenETCS data Dictionary

ETCS terminology refers to “variables” instead of “types”. All the items in the ETCS Language specification are data types definitions and should not be confused as global variables. In spite of this ETCS data type shall be converted into OBU´s internal data types. To avoid confusion by different names for internal and external data types, OBU´s internal types preserve their name.

Internal data types reflect as much as possible the ETCS language also for next improvement and traceability of the ETCS software. An OpenETCS plug-in provides this shared

repository of well-defined data type. Every system engineer takes OpenETCS data type as a reference and creates their own packets. If something is missing they create and specify new data type that can be shared or private. One of the most used primitive data type is the enumeration, which is very easy to use and makes the code really simple to read. Every Developer can import the type and use it in his model but he cannot modify it as well.

3.2.3 Software Design

Once the modeling step has been completed, software design shall be defined to take into account software constraints. The aim is to design the functionalists of SRS to produce functional code, that in the earliest stages it is not based on SIL 4. Design shall cover as much as possible requirements of the SSRS. Next Working Team shall provide a method and a tool chain to produce SIL 4 code. The output of this step is a semi-formal model, which allows producing executable code. According to the requirements, Designer shall define every single block, the complete architecture, the input and output of the sub-system. Each component of the model has a description of the operation modes and behavioral with all the allocated and linked requirements. It also shall provide testable requirements on the target platform.

Translate the model into code

When the model is completed and programmed, the Developers shall check the model errors and un-consistent data or operations. Before generating code, they also shall check all the models and show the linked information. At the end, they produce qualify-cable C code, which shall be:

Readable it shall propagate names and annotation

Portable It shall be independent from the target. The code can be easily being executed on a different machine with a different scheduler

Modular It shall be based on one or several Modules

Just a static memory allocation is required in order to prevent un-deterministic behavioral of the system. The use of a finite execution time duration, is also mandatory (especially for embedded systems), to avoid undefined time of responses and Size optimizations.

3.2.4 Verification and Validation

The **Development Plan** and its relationship with Verification and Validation as quality assurance, project management and configuration management must be clear step-by-step. The guidelines of communication within the Verification and Validation, the authority for resolving issues and approving deliverables must be specified. The development project shall have a predefined well-organized schedule with an orderly flow of material between activities and tasks. Also the responsibilities must be clarified. Every time a verification Team Manager, before starting working, has to be clear on what he/she is actually doing and on what part he/she is verifying. According to the process every step of the V-model shall be validated and verified, from the Requirements until the code. The use of an appropriate technique will help to ensure the consistency of approaches across development teams and may result the best practice solution.

The safety activities in the software development process are closely connected to the verification & validation activities. The collected results documents of the Verification and Validation are used to build the safety activities. These identify unwanted behaviors and re-port them in order to obtain a hazard potential analysis that can lead to the harm.

According to the specification phase the initial risk of a hazard resulting in harm has to be reduced until the development reach the required risk. The Verification and Validation process must verify these specifications and validate them using a complete documentation to show within a safety case that the plan is respected. The software validation has to demonstrate that the developed product meets the safety goals.

Therefore the safety requirements that have been based on potential accidents and accepted risk levels have to be validated. It´s important that the verification and validation

activities work in close iteration with the safety activities. The OpenETCS activities perform the Verification and Validation strategy so the overall approach needs to be defined and planned.

The aim of **Validation activity** is to demonstrate that the objectives of the Validation Plan, which have been set, have been achieved. This concerns the adequacy of the software design documentation and system behavior is complaint with the software requirements.

The input documents of the validation step are the **Components Requirements Specification**, the **Integration Specification**, the **Integration report**, **Test Specification**s and **Test Report.** The output document is the **Software Validation Report**, which verifies the objectives and functionalists from the Requirement Specification.

Since OpenETCS process is based on three main inputs for methodology and product lifecycle: SCRUM[7] methodology, the model Driven Design and the CENELEC software development V cycle, the System Requirements shall be directly translated into formal specifications on which Verification and Validation techniques are applied. The use of formal specification and formal language allows deriving the models using dedicated languages in order to guaranties conversation of properties along the design process. But it is really hard to understand the rules and its correct interpretation. For this reason the model-based testing has been chosen as part of the process OpenETCS.

Formal Method Analysis Strategy

The advantages of Formal methods are pretty well known in terms of requirements specification. For Verification and Validation step they can allow to check the software properties like exceptions, deadlock, run-time errors correctness and consistency. In order to accomplish a V&V process in a very consistent way a formal analysis has to be applied to a formal method.

The most famous Formal Analysis methods are three: Abstract Interpretation, Deductive Verification and Model Checking.

[7] Agile Method, widely described in chapter 5

Abstract Interpretation method creates abstraction of a set of all possible state that may occur there during any execution run. This representation is also called an over-approximation. This interpretation determines particular effects of all the possible concrete behaviors of the program, while the abstraction might lead to consider states that cannot occur in a concrete execution.

Abstract Interpretation determines particular effects of the program relevant for the properties to be analyzed but does not actually execute it. The tools might report warnings related to states that are included in the abstraction but do not correspond to concrete executions.

Deductive Verification methods perform mathematical proofs to establish formally specified properties of a given program, providing evidence. This method is based on the Hoare Logic or axiomatic semantics, in which functions are seen as predicate transformers. In some cases it is too complicated to handle with an automated theorem and must be discarded and do not accept a test case.

Model Checking check all the possible behavioral of a program to determine whether a specified property is satisfied. It is applicable only for program with reasonable state space and the specification are usually temporal properties. A counter counts each time a property is unsatisfied. A tool that fulfills the characteristics required for the V&V in the OpenETCS project is the Classical B, which is partially open-source. A natural language document for the requirements is the starting point of the analysis and the correctness is based on the human check. A model based on specification shall be modeled at this point and the correctness is preserved by proofs.

3.3 OpenETCS Scrum and EN50128

In OpenETCS project proposal the use of Scrum and Agile Development is highlighted as a chance to get a better result for our OpenETCS stakeholders.

It shall be adapted for safety-critical environment based on Safety Integrity Layer 4

(EN50128). The Agile method such Scrum does not naturally fit safety engineering. The main reason is that these kinds of systems require a strict plan, which makes later changes costly. Moreover, safety critical projects have pretty well defined structure, components worker rules but also a lot of traceability and documents are required. OpenETCS shall deal with both of these two problems without losing the benefits [38].

EN50128 and Scrum

Agile Methods like Scrum are well known because they are pretty flexible. Scrum, in fact is able to be adapted to plan, document and specify as it is required in EN50128 for developing safety-critical systems. Scrum is really collaborative and every-one shall learn by the other components of the team in order to obtain a well-distributed knowledge of the tools and Environment. But, EN50128 standard strictly forbade the collaboration inside the team of workers with different rules, such as manager and Validator. Moreover, the project man-ager in Scrum does not exist and nobody such as Product Owner or Scrum Master can take Management Decision but they shall being taken in a collaborative way [38].

Every "stage" [8] can be mapped with the **Scrum Sprint** [9] and all work related with the Quality Assurance is moved outside the Scrum. There is not a lot focus on test documentations in Scrum. To fulfill Scrum with standard, some changes on planning sprints shall be done with a dedicated sprint for documentation or separate activities. The software Quality Assurance Plan, Software Validation and Verification plan, and Software Configuration Management Plan shall be defined up at the starts of the project and maintained throughout all the development life cycle. Every iteration of Scrum consists of a small portion of the project which is extended to all V-model from the requirements analysis until the testing and Verification and Validation. To maintain traceability, a separate session

[8] defined by ISO 12207: "Period within the lifecycle of an entity that relates to the state of its description or realization"

[9] The sprint is a "timeboxed" effort; that is, it is restricted to a specific duration.[14] The duration is fixed in advance for each sprint and is normally between one week and one month, although two weeks is typical.[7]

is reserved and it is also possible to use some tools, which support the process .

The **Scrum Backlog** [10] usually indicates how much resources are needed to accomplish the "stage". It estimates how much time left before the activity is completed and the number of developers needed in this step. In order to adapt it to the Standard a **secondary backlog** should be adopted for safety-critical activities. This safety-critical backlog is an extension of the other one because it contains just non-flexible requirements and it´s really difficult they are changing during the process. The original backlog, instead, is really flexible and adaptable to changing during the process such as requirements. It also preserves some references to the safety-critical backlog and vice versa in order to have a clear communication between each other's. All the process is based on iterations and increments of the final systems. At the end of the iteration the Assessor shall check the work and be able to approve or not the work. If him does not approve the work other iteration on this increment shall be done and the work must be continuously re-planned based on the most recent experience in order to improve the quality of work. A Hazard and Risk analysis shall be done to evaluate the product and give better results. At the end of all the activity a final Sprint shall be done to validate the process. Validator and Verifier as explained in EN50128 are not present in the Scrum schema. The idea is to take a developer, assign him the role of Validator and Verifier in a separate session of the others [38]..

Since the **Project Manager** is not an allowed role in Scrum all the team shall use the backlog to understand the reports for all the steps and make a common decision excluding the ones which are enveloped in that particular step. Validator shall report this to the management, which is also a separate team to the agile method.

Test driven Development is one of the most used technique in that field and fulfill with Scrum. All the developers then shall take into account the testing of the code before the implementation so it enables regression testing and a full documentation of code.

[10]The product backlog is an ordered list of requirements that is maintained for a product.

Chapter 4

Case Study: The ETCS Eurobalise

This chapter presents a case study based on a subsystem of a kernel function of the OpenETCS project. The project team leaders, according to the standards and for a better understanding of the SRS, divided the OpenETCS Kernel into 47 Kernel Functions. One of the first functions has been taken into account is **Manage Location Related Information** (figure 4.1).

The aim of this function is to enable the location of the trackside elements, retrieve the distance between the position of the train and the external trackside devices, support the heterogeneity for a wide variety of location based information and calculate the position by using a reference point. The Information from the trackside can come from Radio or a group of EuroBalises.

The Information received from Radio can come from a National System Authority that allows the movement of the running train or from GSM Antenna sending information regarding the position.

This Function shall memorize the Last Relevant Eurobalise Group Information and calculate the position using this as a Reference Point. The Manage Location Related Information has been split into two parts: Manage EuroBalise Information and Manage Train Position.

The first part has the matter of Performing and decoding the received from a single Eu-

roBalise Message, Collect all the information received from a EuroBalise Group into one message, Check the consistency of the information, Determine the EuroBalise Group Orientation of the Last Relevant EuroBalise Group, Filter the needed information and store all the messages.

The second part shall Validate Data Direction, Calculate Train Position taking the Last Relevant EuroBalise message into account and Manage all the information. After the analysis of the main features of the function, a Block Definition Diagram has been done by ERTMS/ETCS Experts. This diagram expresses the flow of data inside the function. Each sub-function became a part of the main block. At this point, each sub-function has been assigned to a partner that has the scope of developing, describing and interfacing his own part.

Starting from the block Determine EuroBalise Group Orientation, in this chapter, is explained a study case of how to deal with the developing and modeling process.

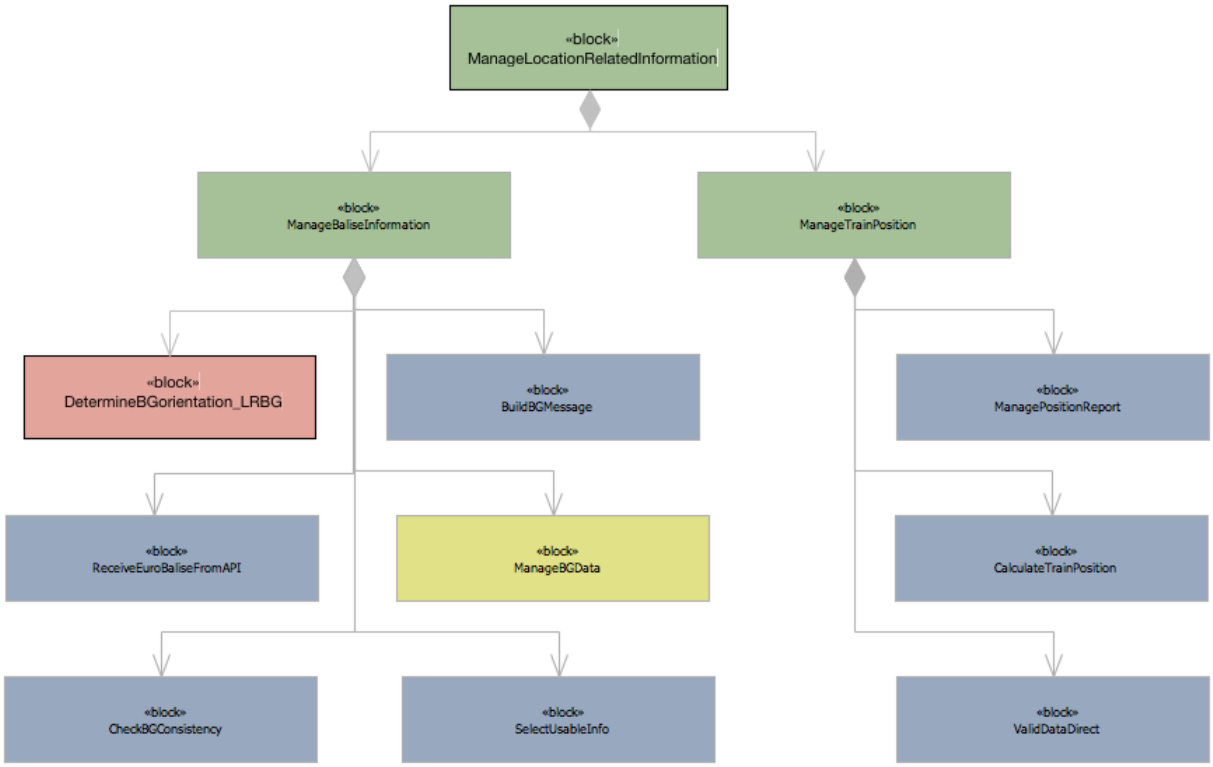


Figure 4.1: BDD: Balise Group Orientation

4.1 Determine EuroBalise Group Orientation

Before analyzing requirements, a starting point is the right understanding of the problem. The aim of this part is to retrieve and determine the orientation of a EuroBalise Group from a received message. An EuroBalise is a trackside device that transmits information concerning position, speed, and direction of the train and track restrictions. Each EuroBalise is grouped into a group composed of a number of balsas between one and eight. As shown in figure 4.2, every EuroBalise has an internal number that refers the position inside the group.

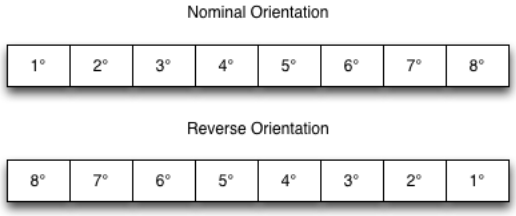


Figure 4.2: EuroBalise Group Orientation

The running train, gets the valise message from the trackside. It receives the message from each single EuroBalise and groups the messages. At this point the train can receive the message in two different ways. The nominal direction means that the train is moving from the first EuroBalise to the others, increasing the internal number. The reverse direction, instead, means that the train is running from the last EuroBalise of the group to the first-one.

This sub-system shall also consider the Radio messages, which have the highest priority and most importance. Also the Level of the ETCS in which the train is running is important, in order to have a better understanding of the checks that it shall do.

4.1.1 Case Study Overview

Figure 4.3 shows the entire overview process of the Case Study referring to CENELEC V-Model and the Model Driven approach.

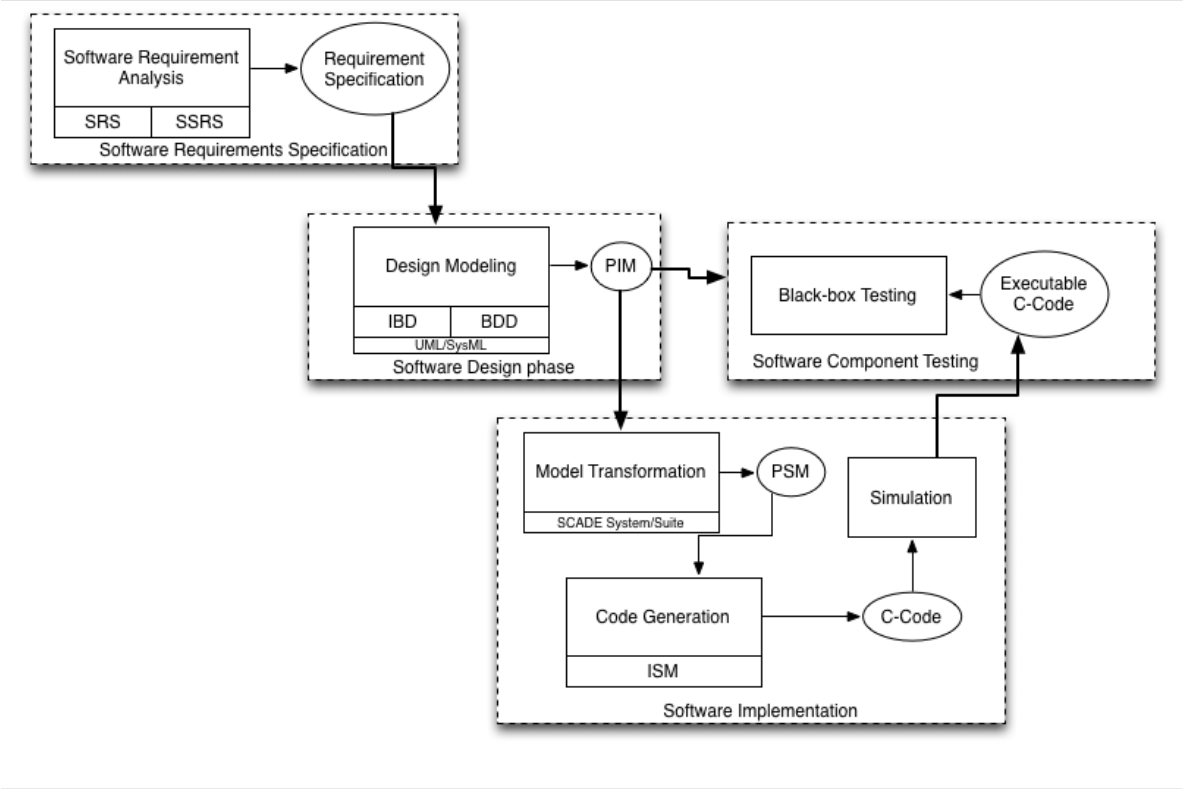


Figure 4.3: Case Study Overview

4.2 Software Requirements Specification

All the software requirements of ERTMS/ETCS are expressed in the System Requirement Specification of the project. Subset 026 is the most important part of the SRS, it contains the biggest part of the system requirements.

The aim of the Requirement Engineer is to discover, trace, qualify, communicate, and have a level of abstraction. In most cases these specifications leave to engineers the freedom for design choices [20].

He has to deal with arbitrary complexity and instant distribution. Each requirement shall be unambiguous, testable or measurable, and necessary for product process acceptability by quality assurance guidelines.

4.2.1 Organize and Store Requirements

The Requirements Engineering team uses RFM ProR to analyze, structure and link requirements. Every requirement is categorized, marked, time stamped, signed by the author, linked to the others, inserted into a hierarchical list and managed. There is also a full traceability between the SRS subset 026 and the ProR List. Requirements Modeling Framework (RMF) is an Eclipse-based platform used for requirement engineering. It uses **ReqIF file**, which is a standard of OMG Object Management Group.

The graphical user Interface of RFM is ProR. ProR is an eclipse plug-in, it is not possible to run it standalone. No special project type is required for ProR, therefore any Eclipse project can contain ProR requirements files. This tool supports the activities found in requirements like traceability. ReqIF allows the structuring of natural language artifacts, supports an arbitrary number of attributes and the creation of attributed links between artifacts. A model ReqIF contains attributed requirements that are connected with attributed links. In ProR a SpecObject represents a requirement, it ha a number of Attribute-Values, and it is organized in Specifications which are hierarchical structures holding SpecHierarchy elements. ReqIF contains a sophisticated data model for Datatypes, support for permission management, facilities for grouping data and hooks for tool extensions.

ReqIF is persisted as XML, and therefore represents a three structure. An eclipse plug-in made for OpenETCS project is available for the team. In this plug-in all the Attributes of the Requirements are yet specified within the ReqIF files where are listed all the requirements, note, and parts of the SRS. It also has the possibility to link requirements to the Block specified in SysML, inside the Modeling step, in order to have traceability between the model and the requirements. At the end of the process every block is linked with his requirements, the system engineer can easily understand how this model shall behave and interact.

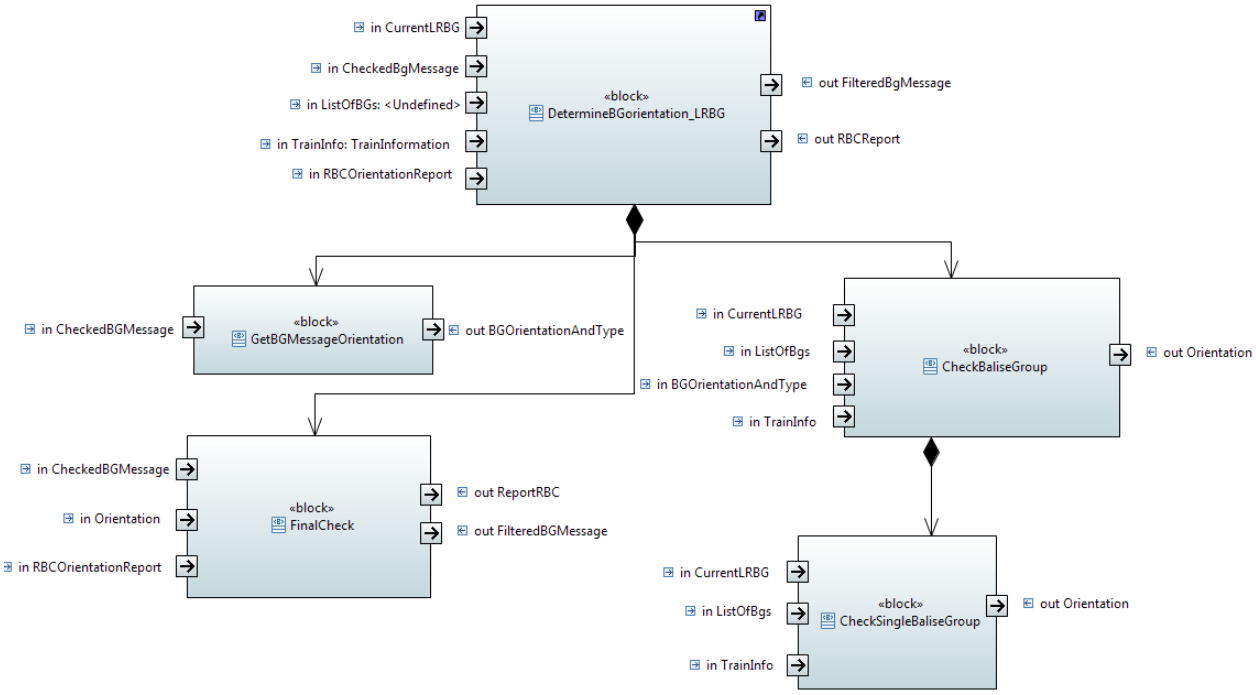


Figure 4.4: Block Definition Diagram Balise Group Orientation

4.3 Software Design phase

The Modeling phase is really crucial for the developing process. It can be split into 2 steps. One is related to the data modeling and one to the architectural/behavioral modeling. The process requires two type of Data models, one for the sub-part internal data type and one related to the exchanged data between sub-parts that must be shared. In spite of this, OpenETCS preserve two different locations, where every partner has to fill with his own models.

Architectural/behavioral modeling step shall have as a result the entire diagram requested by the OpenETCS process document, which are Block Definition Diagram and Internal Definition Diagram.

Data Model must be drown with a **Block Definition Diagram (BDD)** (figure 4.4) inside the SysML environment. Block is a modular unit of structure in SysML that is used to define a type of system component, or item that flows through the system. It may be

subdivided into structural features and behavioral features. The Block Definition Diagram is used to define blocks on terms of their features, and their structure within a system and the relationship with other blocks.

Usually the block refers to reusable part of the system in others. In this type of Diagram also some properties of the block are specified, for example inputs and outputs of the blocks and the hierarchical tree of them, which specify the container, and content of blocks. An arrow specifies the relationship between blocks but also components and containers. These relationships are of two types; they are called Aggregation and Composition.

Aggregation implies a relationship where the child can exist independently of his parent instead the Composition implies a relationship where the child cannot without the parent. In safety-critical environment especially in embedded system the use of Aggregated relationship is strictly not recommended. In fact Scade System, which has the same platform of SysML does not allow the users to use it. The **Internal Block Diagram (IBD)** is a traditional System Block Diagram and shows the connection between parts of a block.

The frame of an Internal Block Diagram contains sub-parts of the blocks, and flows ports between parts. IBD for a good readability should not contain sub-parts of the components.

IBD and BDD (figures 4.4 and 4.5)in Model Driven Development represent the Platform Independent Model (PIM) for their independency from the specific platform used.

4.4 Software Implementation

The SCADE Family Tool allows users to pass easily from an Architectural Model made out of Blocks to an Implementation Model by Using SCADE System. The architectural Model Defined in **Papyrus-SysML** can be easily imported into SCADE System since SCADE System uses the same platform (Papyrus).

The generated UML files can be also easily imported into SCADE, but sometimes some conflicts can happen between different versions of Papyrus. In spite of this the users shall

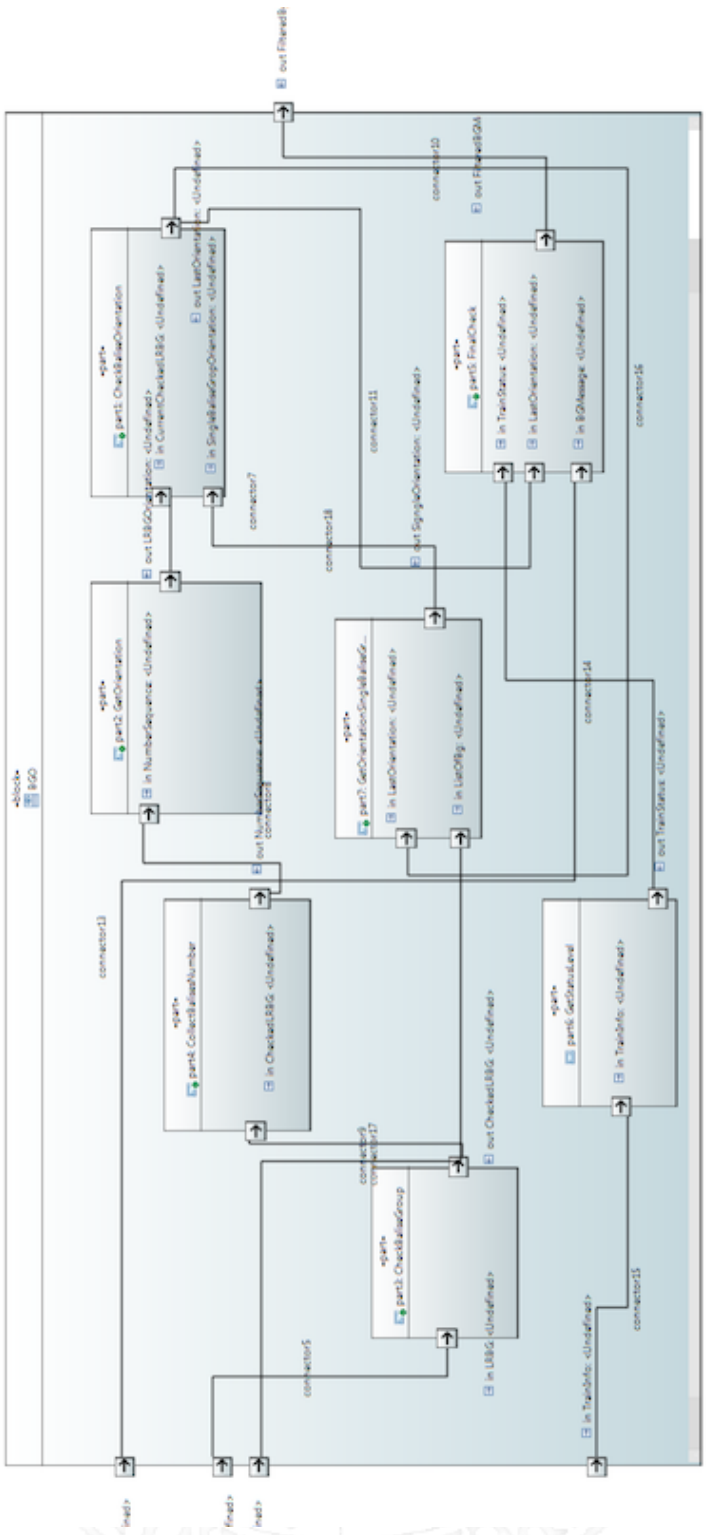


Figure 4.5: Internal Block Diagram Balise Group Orientation

rearrange the draw by hand. Also some of the linking of the Exchanged Data between blocks can be skipped when SCADE System try to import the models.

However, Scade System is able to import the architectural model and the file that it generates can be easily imported into SCADE Suite in order to obtain an Implementation Model. The user can decide to implement each block if it is strictly a part of the main model or leave the block un-implemented because the block refers to interfaces, API or external agents.

Moreover, SCADE System does not support some types or link between blocks present into SysML because it is used just for implementing embedded System. At the end into SCADE Suite Users will find the implementation Model made out of operator. The operator is a kind of block built for an easily understanding of the product and for having information hiding.

Every block shall be programmed and programmers shall take care of the exchanged data between blocks. In figure 4.6, It is shown the architectural model of the Software component under development, in model driven architecture is called **Platform Specific Model**.

Implementation Specific Models, are shown in figures 4.7 and 4.8, both of them are two different parts of BGO. The first one is composed of a state machines, and the second one of general logical, arithmetical and structural operations.

4.4.1 Simulation

SCADE provide also a simulator in order to verify the model with a debugging, a functional testing, a breakpoints and stop conditions it is also able to record and play test scenarios. Simulator enables the simulation and debugging of SCADE models by executing C code which is transparently generated by Code Generator.Simulator has a graphical interface where it is possible to follow the propagation of the data from the Inputs to the Outputs. It is also possible to setup the cycle time and the changes of stimulus during the

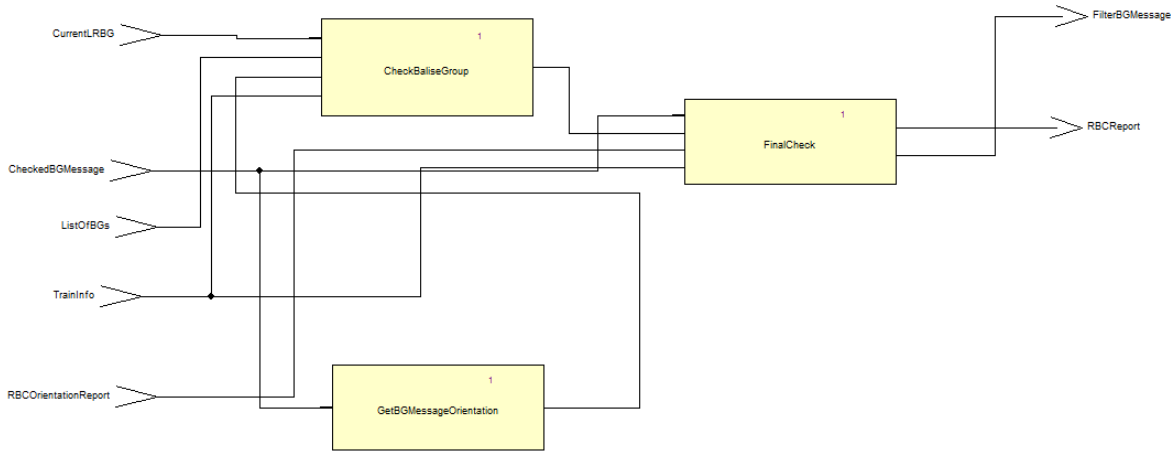


Figure 4.6: SCADE Balise Group Orientation Platform

evolution of the time. Simulator creates also executable files that runs over windows shall and a set of command specialized for the testing.

4.5 Software Component Testing

In order to obtain an automatic test cases generations the Process OpenETCS uses MaTeLo. Matelo 5.1 is integrated in Eclipse Juno and can also be easily integrated in the tool chain OpenETCS. This tool shall be part of the tool group T2 as explained in EN50128. The earliest stage to generate test cases is to draw a graph, which is a Markov Chain, composed of state and transitions.

States refers to all the possible internal state of the System Under Test and the transitions refers to the several operation that the System under Test can accomplish or not. Every Markov chain has a hierarchical structure, a starting point and an end point.

Every state shall be wired and shall be at least on path from the starting point to the end. Every state, excepting for start and stop, can be composed of other sub chain. Drawing of the graph is not simple, the Engineers shall think about all the possible state and operation that the System Under Test can reach and the entire possible path to reach the final state.

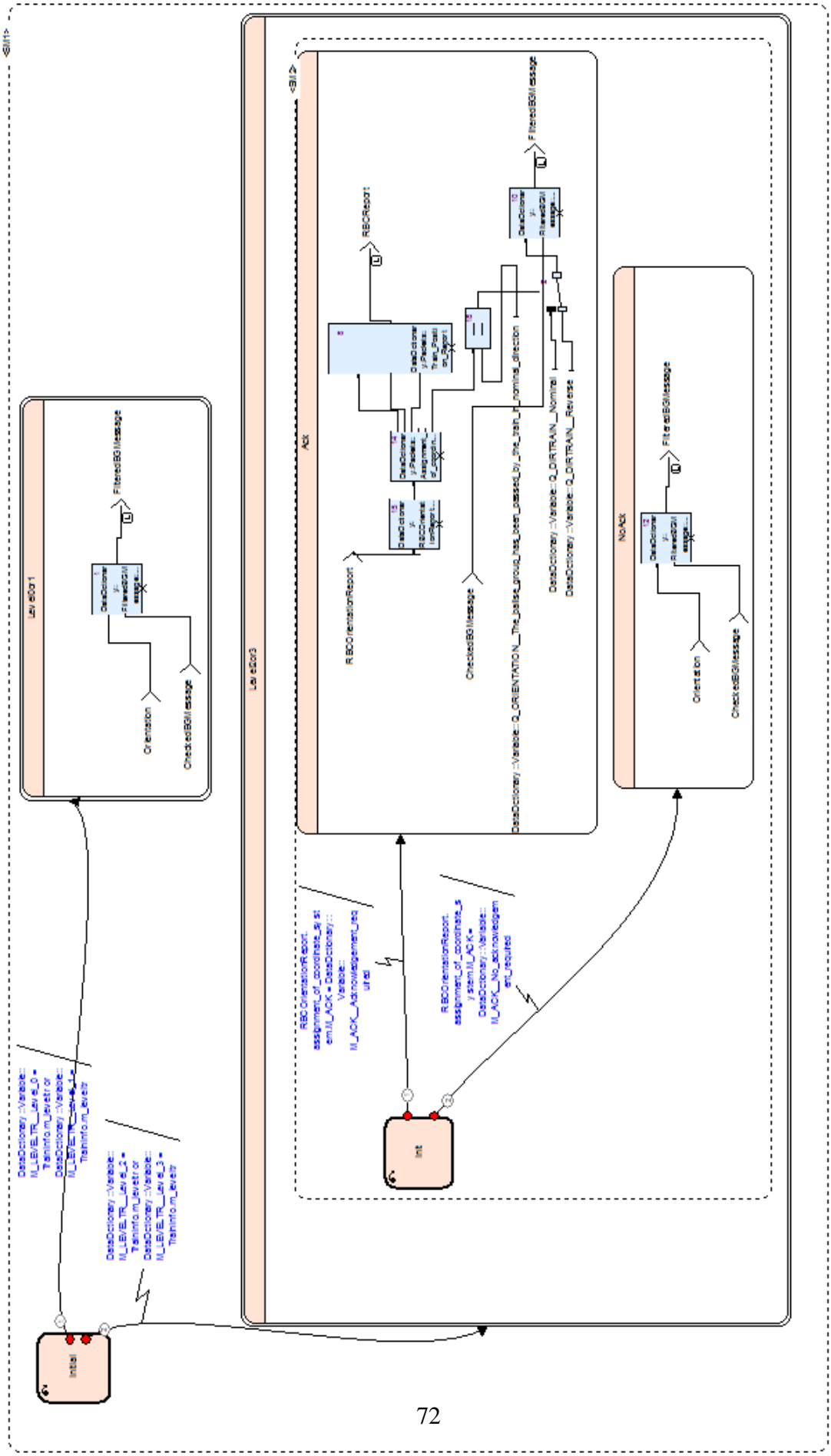


Figure 4.7: SCADE Block: Final Check

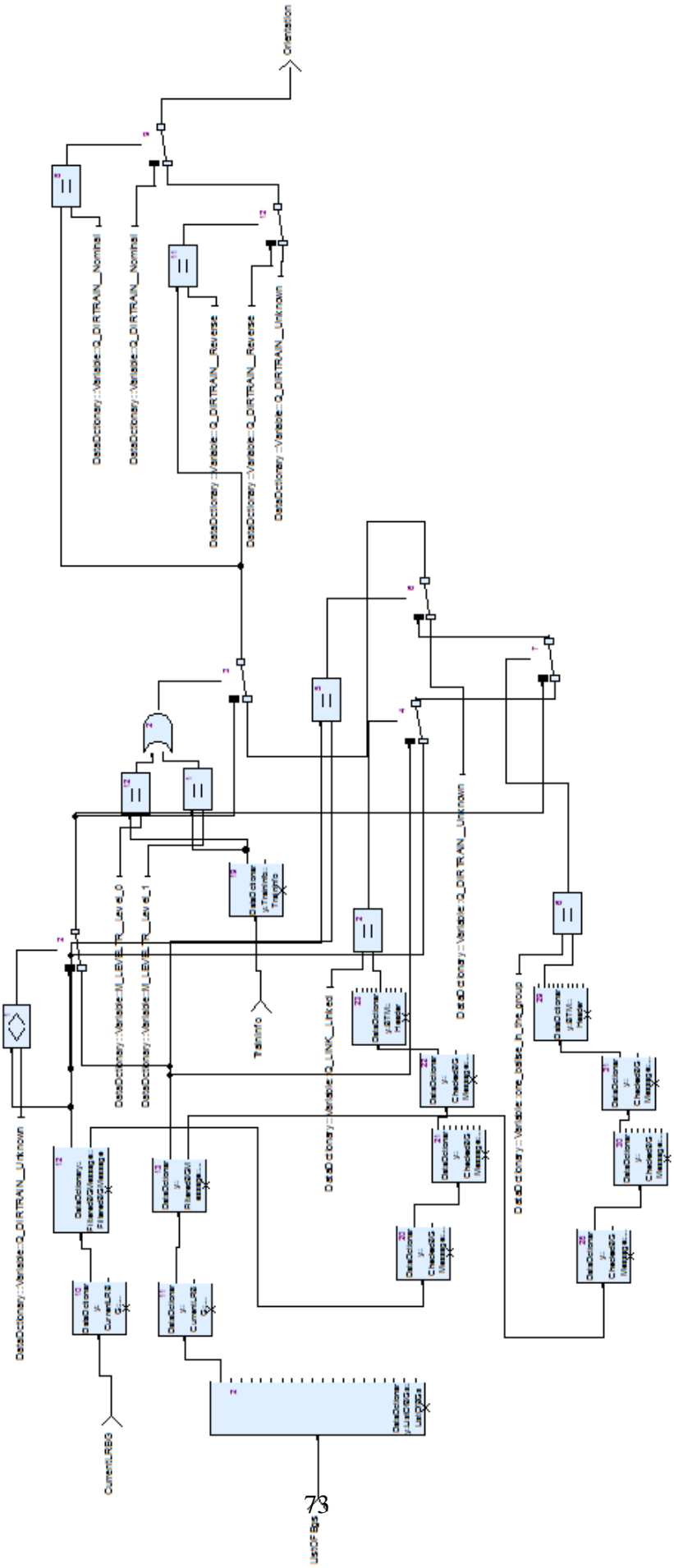


Figure 4.8: SCADE Block: Check Balise Group

Requirements traceability

Requirements can be imported into the tools in different formats. Reqif file generated from Doors are fully compatible with the tool and the entire field present into ProR are visible. To obtain a good traceability every requirement can be linked on the transition where it is propagated and written on the Test reports at the end of the Process. The requirements at this point are also used to create and understand the main model and the System under test in order to drown a consistent Markov chain.

The test cases traceability generated is maintained into a script file with the requirements as a comment. When the tests are executed an html file (or other formats) is generated. This file contains all the traceability of the Verification and Validation on the System Under Test. Engineers, at this point of time, shall analyze manually all the reports, one by one, and check if there are un-consistency, mistakes errors checking the inputs, the outputs and the requirements.

Every report has a check box on the right where engineers can check when they reveal a failure of the System Under Test.

Input Data Types

Every test case shall be different from the others in terms of different input. All the possible combination of input can be too many and the most part of them is not useful for the Verification and Validation step.

Just few Input changes the state of the system, which the engineers in fact shall, refers too. Systems under units can be of different languages and natures with a lot of different data type definition. Matelo allows creating data-type definitions as structures, enumeration, integer or string. New data then can be generated and initialized with an existing definition or a custom one.

Data shall be strictly linked on the arrows of the Markov Chain this means that this data is used as an input of new test cases. It is also possible to create a combination of values

of a type or for integer create a variable with a incremental value.

Each of this value creates a new test case to submit to SUT. In the definition of the data it is possible to specify some restriction of the values as Upper or Lower Bound and the step of the incremental values, The consistency of the output at the end shall be checked, the tool in fact creates a new error if the output does not respect the rules specified by the data definition.

Script and Operations

The transitions of the Markov Chain refer to a set of Operation on the system. These operations can be created b new editing the Script File. Since, the System Under Unit is a Scade Model, the script file is of the format SSS, which is an Esterel property. When a simulation is running into Scade Suite the tool creates some executable file for each operator inside the project of Scade.

These files are executable by using the Windows shell. Scade Simulator provide several main faction to run the model from the Shell, the most three important are SSM::set, SSM::cycle and SSM::get. The first one is used to set the input of the SUT, the second one to run for just one cycle the simulation and the last one to take all the results in order to have a traceability.

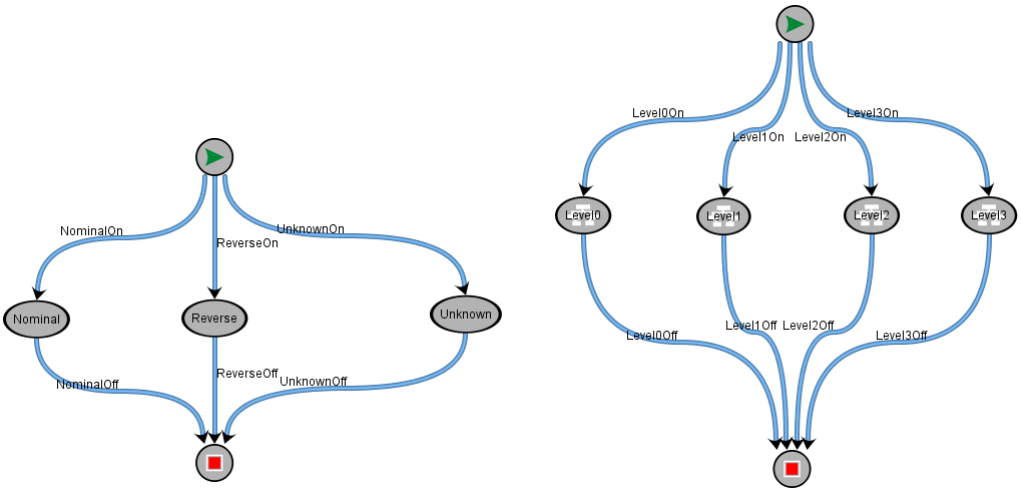


Figure 4.9: Matelo Markov Chain

```
1 SSM::set DetermineBGOrientation_LRBG::DetermineBGOrientation_LRBG/
    CheckedBGMessage.headerflag.0.header.n_total DetermineBGOrientation
    _LRBG::DataDctionary::Variable::one_balise_in_the_group #
2 SSM::cycle
3 ### Check ##
4 SSM::cycle
5 set testout [SSM::get DetermineBGOrientation_LRBG::
    DetermineBGOrientation_LRBG/FilterBGMessage.q_dirlrbg] if {
    $testout != "DetermineBGOrientation_LRBG::DataDctionary::Variable::
    Q_DIRTRAIN__Reverse" } {      puts $fd "error : Test Case: Test
    Case2 : value ( $testout ) different from expected value ( Q_
    DIRTRAIN__Reverse )" } #
```

For all the transitions these three functions shall be executed to submit the stimulus, run the system and take the results.

It is possible then to create a sequence of operations and use it for all the transitions. SSM::set and SSM::get use different parameters of different natures to specify the path of the Input or Output and the value. For this reason Matelo allows the use of different parameters.

Final Stage and Executions

After the defining of the Tests, a test cases script shall be generated. Matelo generates several test cases in an automatic way referring to the Markov Chains. The tool tries to make a full coverage of all the paths of the Markov chain graph by generating different combinations and using different data.

Tool checks all the graphs, where every state shall be linked and the final stage can be reached. At this point it is possible to execute the script by using the primitive SSS - scenario “path of the script”. All the tests are executed and the results reported by means of the shell command SSS.

At this point of time a Validation Report can be automatically generated, it shows the oracle’s inputs, the expected outputs and the actual outputs. For each one of this there are the requirements linked. The engineers shall select or unselect the check-button indicating that the actual output is respecting the requirements.

Chapter 5

Process, Languages and tools evaluation

As software becomes more and more important to all aspects of industry, we need to adopt the best practice in order to improve the quality of the processes in use, and therefore achieve targets relating to time, budget and quality. For that reason, evaluating the process and benchmarking the tools can improve the process of software development and may increase the cost reduction, improve the quality and reduce the time-to-market. New methods and tools continue to proliferate without supporting evidence as to their benefits over existing approaches, indeed it is surprising how little effort is directed towards evaluating methods and tools.

There are two main types of evaluation methodology:

- **quantitative evaluations** to establish measurable effects of using a method/tool
- **qualitative evaluations** to establish methods and/or tools appropriateness for example how well a method/tool fits the needs and culture of an organization

Quantitative evaluation and its measurable effects are based on observed changes in production, rework and maintenance time or costs, identifying the benefits of a new tool or method in terms of real measures [22].

Qualitative evaluation is strictly based on the technical and specific needs of the project considering what a tool or a method must ensure and produce.

There are also hybrid methods that involve both the subject and an objective element.

There are three main ways of organizing an evaluation technique:

- as a **formal experiment** where it is requested to perform a task using different methods or tools under investigation. It can be investigated using standard statistical techniques.
- as a **case study** where it is tried out on a real project using the standard project development procedures
- as a **survey** where organizations or engineers skilled in specific methods or tools are asked to provide information about the same.

The methods and tool benchmarking for the Project shall highlight the strong points and weak point of the potential languages or tools. Each partner need to use the same tool chain, therefore a tool comparison and a straight selection is required. A formal model is composed of two different aspects: proof aspect and modeling aspect [22].

- **Modeling aspect** is required to describe all the aspect of the SRS SUBSET 026
- **Proof aspect** is required to show the safety properties and verify the capability of the method or tool

For the modeling aspect the idea is to cover all the different means of description needed for the SRS, in order to show the strong and weak points of a potential language/semantic. Moreover, there is the need of having a sufficient and self content part of a functionality for the proof aspect on order to verify the proof capabilities of the method/tool [22].

Due to the links between languages, tools evaluation of language and tools a single benchmark is required for the evaluation process.

The benchmark process is composed of two phases :

- **Initial benchmark** during which there is a proposed methodology, tools and/or developed a case study [22].

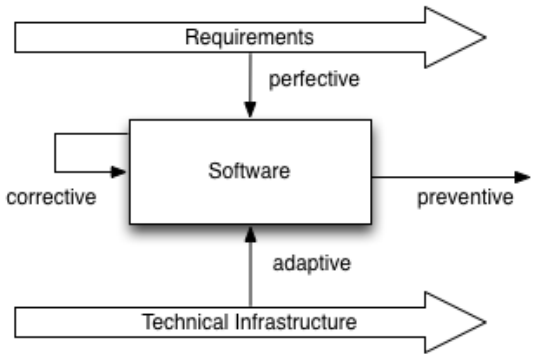


Figure 5.1: Maintenance activities

- **Assessment benchmark** where it is assessed the initial proposal. Assessors analyze the proposed means and/or tools to complete the pages on tools and criteria to propose other models. Assessors also guarantee the quality of the process [22].

5.1 Model Driven Engineering Evaluation

The goal of Model Driven Engineering is to increase the quality and the speed of software development using a higher level of abstraction, modeling techniques of transformations and code generation. That fit perfectly with the Requirements of CENELEC but there are some weakness to face to overcome them. A particular focus is on the maintenance and enhancement of the application as well as model transformation languages templates and tools

Maintenance

Maintenance is a crucial aspect, based of four types of activities: adaptive, corrective, perfective and preventive maintenance as it is shown in figure 5.1.

- **Adaptive Maintenance:** the system has to be adapted to a changed technical environment

- **Perfective Maintenance:** the system has to be improved of the functionalities

matching the redefinition of requirements

- **Corrective Maintenance:** the errors shall be detected and have to be fixed
- **Preventive Maintenance:** actions taken to avoid difficulties in maintenance activity and in the future.

Maintenance, is so a strong point of the Model Driven Engineering because with this approach there is a significant reduction of time. Model Driven, in fact, keep model, requirements and code heavily paired. For this reason if a future modification of code is required it directly gives an effort to the requirements and vice-versa.

Modeling languages

One of the main problems of the UML or SysML language is that it does not contain enough information about the meaning of the modeling elements (a UML class is just interpreted as a plain programming languages class). The biggest part of the modeling tools provides a direct mapping between models and code, for that reason the model and the code have the same level of abstraction.

To resolve this, a good idea is to use an expressive dictionary between PIM and PSM models. In fact, openETCS uses the so called "Data Dictionary" (chapter 4.2.2). For some specific domain the OMG provides UML profiles that specify parts of the PIM and PSM. Tool vendors often define their own stereotypes, but in general cases this are not enough to accomplish the model.

A carefully tested and proven code, as part of a set of templates can be the solution for this problem. The reuse of model or code is a key benefit of this method, and in some cases it is possible to use the most of the just the effectives templates. This way, PIM using stereotypes can be converted to executable code. The custom UML profile has several additional benefits: the UML profile can be adapted to the needs, developers do not have to lear a design vocabulary and existing knowledge and experience are really used.

Class names, attributes names, data type names and other information in the code are replaced by variables and control structures. During code generation the variables are replaced by the corresponding values that are defined in the UML models. The stereotypes in the PIM are interpreted and transformed to other model elements in a PSM.

A **reference architecture** is a set of reusable structures, used in the development process that can change over time. This means that several versioning problems can happen with the references. MDA aims is to reduce this issues with two approaches: adopting a new architecture with all the improvements and where the changes are reflected into the UML stereotypes and in the code generation templates, and using an appropriate process with a configuration management.

Change Requirements or New Requirements is an other crucial aspect of the MDA, the problem is of maintenance cost in terms of time and risk. OpenETCS resolves this problem with an embedded plug-in that maintain a full traceability between requirements, models and code. This may represent an huge effort in the process.

Dependency of Tool Chains

Upgrades or changes in the run-time environment are frequently enough to take into consideration. Operating system, GUI framework, or a single version of a tool inside the tool chain may cause a lot of issues. Due to service level contracts, diminishing support for older versions, or new components that need the newer version, the user of a technology often has to follow the upgrades.

Therefore it is needed to analyze and keep track of dependency chain of each system. This analysis includes the component of the system, the version of the tool and any third-party components as well as compilers and the whole development tool chain. Upgrades of a tools might allow changes in the system or even force certain changes to the system. For example if someone makes some modifications to the model with a new version tool, everyone must update his own tool.

Conclusion

Model Driven Architecture, is one of the increasingly methodologies of future, it separates the business logic from the technological aspects. Unfortunately, there are some dependencies to face between the toolchain, the modeling methods and the languages. All these elements have their own innovation cycle, for that reason an unique and standardized toolchain may represent the solution. Also the OMG MDA specification is underspecified and it is required a standardization of the whole process, especially regarding the translation between PIM and PSM.

This can make software engineering able to move a big step forward.

5.2 OpenETCS Improvements

OpenETCS, with his process, gives a method to resolve and accomplish the high complexity of building safety critical system using three main principles: CENELEC EN50128, Model Driven Engineering and Agile Method Scrum.

Starting from this, considering also the future development and future needs of the European Railway, the following chapter lists some possible improvement of the process, toolchain, used tools, modeling and methodologies.

Moreover, OpenETCS process is strictly based on three main sub-processes:

- **Develop a Toolchain**
- **Safety Process**
- **Regular Process**

The actual process runs in parallel all these, for that reason it may keep confusion between developers, tools version controlling problems, or mistakes. For that reasons a best approach can be: to develop before a performant, well defined, consolidate toolchain using

a case study as a reference. With this toolchain Developers have a stable platform where they can start working regularly.

After this, distinguish a safety process from a standard process may cause some waste of time in terms of clarify what is under development. A direct and unique safety process can make the progress slow for a very early stage and considering the average quite performant. The Idea is to marge into the safety process the regular process too.

Avoid to use Agile Methods such as Scrum within safety critical environment and also have an unique and clear schedule imposed by CENELEC EN50128 V-Model may give an effort to the understandability of the whole team.

The toolchain provides the tool support and the development process to provide a formalized specification of RSR and un executable code of the OBU. The toolchain platform hosting the tool chain is Eclipse with the Eclipse modeling framework (EFM). This implies that the toolchain will be a set of Eclipse plug-in and we can rely on already available plug-in and features for the versioning. The use of EFM will also assist the software development, by providing a meta model and API for manipulation EMF components.

Unfortunately, SCADE system does not allow the integration within Eclipse framework, in spite of this a second toolchain by esterel is required to obtain code generation. Moreover, V&V process, is not yet integrated inside the Eclipse platform and this may cause some issues to the developers.

In order to improve the toolchain, showed in figure 5.2, some plugin should be created to obtain a complete and full traceability and a flexible interoperability.

For that reason OpenETCS should develop:

- a plug-in that generates retroactive link from model to requirements, to improve understandability of the System under development
- a plug-in to make the toolchain able to share data from the Eclipse toolchain to SCADE automatically

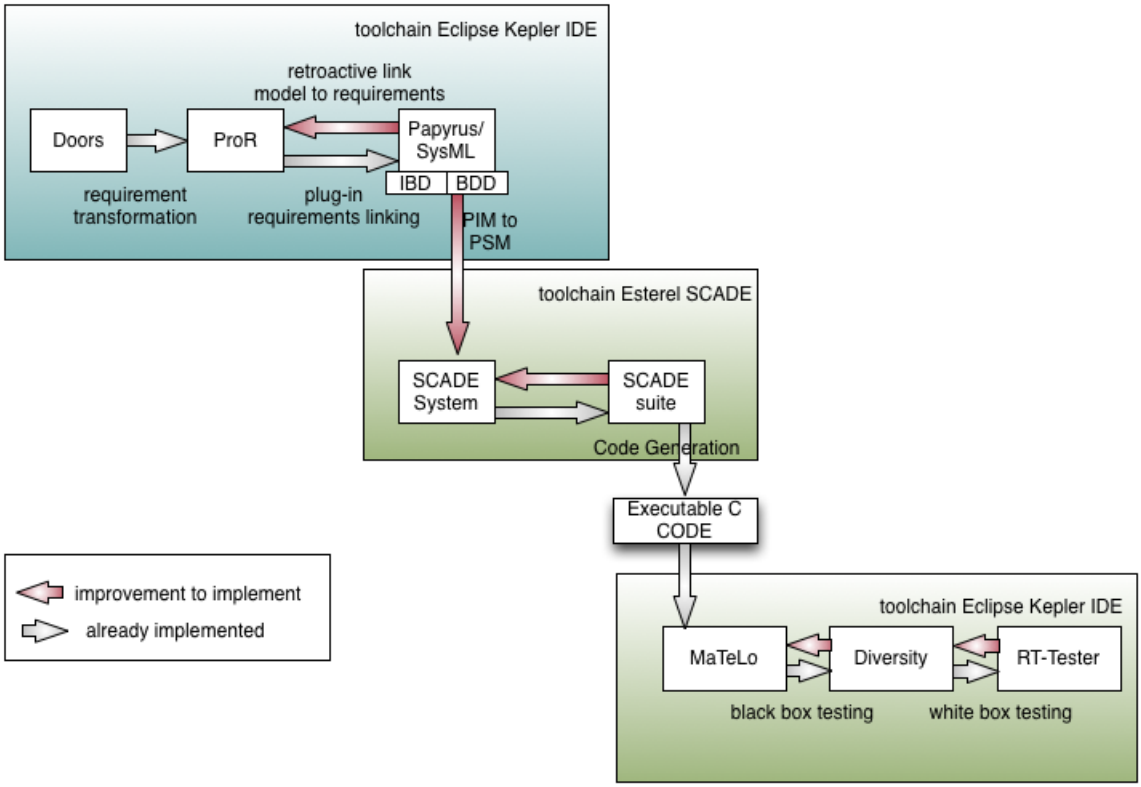


Figure 5.2: OpenETCS toolchain

To make the toolchain flexible and easy to maintained it is required an interface between tools without loss of informations. The interoperability mechanism should assist the flexibility as well as give some guidance to integrate new tools along the tool chain development. Each tools should be agreed with the others for a common format, helping the link the activities altogether and ease the tools update or change. It keeps the tools independent each other.

ModelicaML

A significant change, could be the adoption of a new tool that can generate executable code and uses strictly the toolchain Eclipse. This tool takes as input the SysML files of Internal Block Diagram and Block Definition Diagram, define his own model and transform it into C executable code.

A possible tool that can fit these requirement can be **ModelicaML**.

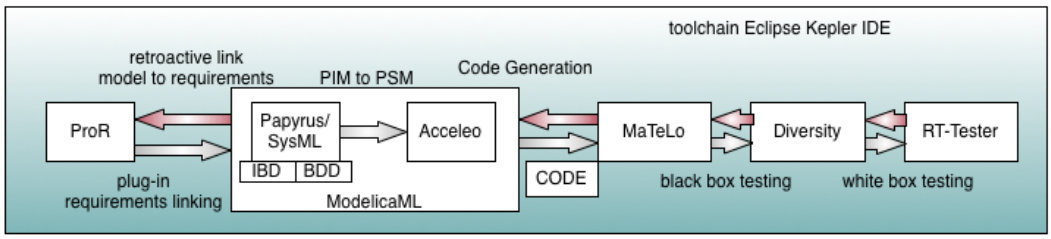


Figure 5.3: OpenETCS toolchain possible improvements

ModelicaML is a graphical modeling tool for the description of time-continuous and time-discrete/event based system dynamics. It is an extension of the OMG UML and enables the generation of executable Modelica[1] code. ModelicaML extends the graphical modeling capabilities of Modelica by providing mode diagrams (UML based) and the ModelicaML profile is an extension of the UML meta-model it can be used as an extension for both UML and SysML It, also, support the requirement organization compatible with ProR. It is supported as a tool easy to integrate inside Eclipse Platform. Therefore, ModelicaML uses a subset of UML, extends the UML meta-model with new constructs in order to introduce missing Modelica concepts and reuses some concepts fro SysML. It generates from graphical models Modelica code. Hence, the ModelicaML execution semantics are defined by the Modelica language and ultimately by a Modelica compiler that will translate the generated Modelica code into an executable form. Papyrus UML can be used as modeling tool and extended by the ModelicaML profile and customized modeling tool features (i.e. diagram selections or dedicated toolbars).

The ModelicaML code generator uses **Acceleo**[2] Eclipse plug-in which is MDA approach based and the model-to-text recommendations of the OMG.

Unfortunately, ModelicaML and his code generator is not certified for CENELEC EN50128 standard. That is a crucial aspect that shall be solved with particular care.

To deal with this issues it is required to make some changes in the process and improve-

[1]Modelica is an object-oriented, declarative, multi-domain a-causal modeling language for component-oriented modeling of complex systems.

[2]Acceleo is a pragmatic implementation of the Object Management Group (OMG) MOF Model to Text Language (MTL) standard.

ment.

First of all a **significant improvement of the data dictionary** (fig. figure 5.3) can give huge effort. Indeed, Data Dictionary should not contain data declaration but also the behavior and code of the more used component of the ETCS (i.e. Radio Management, Eurobalise decoder etc.). For each component, in-fact, a specific behavior is expected, for that reason all the implementing code should be reported with a full documentation in order to improve the reuse of code. At the end a Domain Specific Dictionary shall be created that support the development process with its specific languages made out of ETCS components.

ModelicaML is not able to generate C code certified for CENELEC EN50128 and create these specific domain components. To resolve this issues **SCADE can support the early stage of the developing phase** outside of the toolchain, using the generated code inside **Xtent**[3] that support the code integration inside the tool.

Polarsys

A new open source industry collaboration, called **Polarsys**, is being created by the Eclipse Foundation to focus on building and maintaining tools for safety critical and embedded system development. Partners from all over the Europe are involved in that project and it is still open to any organization interested in participating. The goals of the project are:

- build a platform software that can support the development of embedded safety-critical embedded system
- advanced certification of their development tools to adhere to government and international regulations.
- support full development lifecycle, from requirements through to system verification deployment and in-field support and maintenance

[3] Xtext is a framework for development of programming languages and domain specific languages.

- They must last for a very long time, in some cases 30-70 years

Polarsys supports Requirement analysis, System/Design engineering, SW/HW Development, Quality, Configuration Management, Test /Verification and Validation, Maintenance, Process, Field Engineering and Monitoring.

For each one of these supported phases there are different tools:

- **Modeling:** Papyrus, Sirius
- **Compiler:** GCC/clang
- **Code Generator:** Acceleo
- **Debugger:** GDB
- **Static Analysis:** Frama-C
- **Simulation /Emulation:** QEMU
- **Integrated Development Environment:** Eclipse

Sirius enables the specification of a modeling workbench in terms of graphical, table or tree editors with validation rules and actions using declarative descriptions. All shape characteristics and behaviors can be easily configured with a minimum technical knowledge. This description is dynamically interpreted to materialize the workbench within the Eclipse IDE. No code generation is involved, the specifier of the workbench can have instant feedback while adapting the description.

Sirius and Papyrus support the Acceleo to improve and automatize the code generation phase.

5.3 Agile Scrum for Safety: Advantages and Disadvantages

Safety-Critical system developer plan is usually well specified before the start, it is not flexible so changes and improvement may cause some decease inside the process. Scrum helps in that way to give an effort to adaptability. In fact, there are some different and continuous feedbacks by the customers, and the Assessors. It is easy to make next improvement based on the previous experiences. Assessor is enveloped in the process from the earliest stage and gives a huge effort to the team. Moreover, Scrum helps the developer team to better understand the resources and the time to accomplish the project at all stages and to map all the requirements and functional imposed by the standard.

All the role in safety-critical environment are well defined, this makes the software developed and the process really reliable. Scrum, instead, confuses the roles and mistakes are possible.

Every time a decision shall be taken a grooming is needed, considering also the daily grooming and the end sprint grooming. It can cause a waste of time. On the long run all the advantages can be consider un-useful. The team at this point can decide to make two meetings/week instead of daily scrum (24h) but in this way all the benefits of the project are really un-appreciable.

The more safe the software shall be the more constraints are present into the process. For that reason applying Agile Methods to a safety-critical process does not make the things easier indeed. The sprints, in the case of Scrum, shall be done several times also to improve the documentation. For that reason all the developers may get confused after few iteration because they are not having a wide vision of the process. Moreover, make different things at the same time may cause confusion in the organization and the planning of the project.

For that reasons large companies, with a lot of skilled personnel prefers to use not Agile Methods to develop heavy certified and reliable software.

5.4 Tools benchmarking

Following the criteria expressed before, a tool benchmarking and evaluation is required in order to have a good assessment and rating on different tool for each section: Requirement Analysis, design and modeling, V&V and actual coding. All the tools benchmarked shall be evaluated with an **hybrid approach** using a **formal experiment** as a reference.

Tools are eliminated if they do not fit certain firm requirements.

Each one of the following tools are compared also on the base of:

- key capabilities
- target languages
- output (proof and code)
- manual or automated use of the tool
- integration in the tool chain and development process
- Expertise level (prerequisite needed)
- Underlying technologies
- Traceability between source documents and deliverables
- Certification SIL-rated or existing industrial usage

Follow several examples of benchmarking and tool evaluation.

5.4.1 Papyrus/SysML

Key benefits:

- User-friendly tools for graphical language

- Different views of a system: Requirement, logical structure, behavior, physical structure etc
- Modeling the SRS functions structure with SysML improves the cooperation of how the function interact
- showing good performance in terms of looks stable and responsiveness of the User Interface
- High compatibility with other tools

Expertise Level:

Basic knowledge of UML/SysML modeling

Target Languages:

Model, XMI format, Text artifacts

Toolchain and Underlying technologies:

It is possible to include inside toolchain

Certification:

Used in several industrial environment, not certified for text artifact. Possible to use as a (T1 class EN50128)

Main Restrictions:

- Semantic not strong enough: language has to be adapted for the standards
- All the model information cannot stand only on diagrams
- Difficult to hide/show parts of a model

- Versioning Control
- Browsing through model diagram could be better
- Creating manually the requirements takes too long and produces too many diagrams

5.4.2 SCADE

Key benefits:

- The model is strictly formal and concrete
- The model is executable and verifiable
- The model correspond to the implementation
- Code generator qualified for safety-related software development compliant to CENELEC EN50128/SIL 4
- Tools for requirement tracing, model test coverage, report generation

Target Languages:

C-code, ADA

Expertise Level:

Advanced knowledge, It is recommended a 5-days course

Toolchain and Underlying technologies:

It is not possible to include into toolchain. Standalone software.

Certification:

SCADE is targeted for safety related software. Therefore it is code generator is validated and certified for these purposes, and all tools available that are required in those development processes.

Qualified for CENELEC 50128 SIL 4 or other DO-178 B.

Stable version:

SCADE Suite 6.4

Main Restrictions:

- The formalized specification derives from textual specification structure and model structure which is not an optimal implementation
- For mode complex system or more abstract, less formal intermediate layer language like SysML reasonable between textual specification and SCADE

5.4.3 MaTeLo

Key benefits:

- Automated requirement tracing based on SysML test models
- Fully automated test cases, test data and test procedure generation for complex concurrent real-time models
- Justified test strategies compliant with standards
- Automated Software Testing for SCADE software
- Proven product for several domains
- Open Interfaces (Doors, ReqIF format)

Target Languages:

Batch scripting, No target languages

Expertise Level:

Advanced knowledge, It is recommended a 3-days course

Certification:

Not Certified for standard EN50128, it is possible to use as T2 tool. Used in several safety-critical environment.

Toolchain and Underlying technologies:

It is possible to include inside toolchain

Stable version:

All4Tec MaTeLo 4.7.8

Main Restrictions:

- It is not open-source
- Hard to create particular Test cases for data Type Enumeration for Data Dictionary format
- The current version has bugs, It is not stable

5.4.4 RT-Tester

Key benefits:

- Automated requirement tracing based on SysML test models

- Fully automated test cases, test data and test procedure generation for complex concurrent real-time models
- Justified test strategies compliant with standards
- Tool qualification according with the standard CENELEC EN50128
- Automated Software Testing for SCADE software
- Proven product for several domains
- Open Interfaces (Doors, PRoR)

Target Languages:

No target languages

Expertise Level:

Advanced knowledge

Certification:

Not Certified for standard EN50128, it is possible to use as T2 tool. Used in several safety-critical environment.

Toolchain and Underlying technologies:

It is possible to include inside toolchain. Eclipse.

Main Restrictions:

- It is not open-source
- Hard to integrate with other software: Use a different SysML format

5.4.5 Criteria of the Evaluation process

One of the most important steps of evaluating and benchmarking a method or a process is to clarify the criteria that shall be used.

Criteria in regards of CENELEC EN50128, are explained below for each significant phase of the project:

- **Criteria for Software Requirement Specification**: formal methods, modeling, structured methodology, decision table [10].
- **Criteria for Software Architecture:** defensive approach, fault detection & diagnostic, error detecting code, failure assertion programming, diverse programming, memorizing executed cases, software error detection, effect analysis, information encapsulation, fully defined interface, formal methods, modeling supported by design, structured methodology [10].
- **Criteria for Software design and implementation**: formal methods, modeling, modular approach, components, design and coding standards, strongly typed programming language[10].

Criteria in regards of OpenETCS project, are explained below:

For the Modeling, Verification and Validation aspect, openETCS shows all the Highlight points of the SUBSET-026 that are used as a base for the benchmark, in order to understand what the tool shall be able to achieve:

- **State Machine** huge state machines, contain a chart with time, which should be not too long
- **Arithmetics and Braking curves** The OBU must calculate several braking curves to determine if it is not exceeding the safe speed / distance. Indeed it is not always possible to do it in the high level language, but rather in low level language like C or ADA.

- **Truth Tables and Logical Statements** SUBSET-026 can be considered as a tool box and there is also a lot of modes / information / functionality available. All these possibilities are combined into big truth tables representing hundreds of cases. The modeling of these tables will indicate if the review of this is easy or not according to the SRS.

- **Data structure** SUBSET-026 defines the format and content of messages for ERTMS/ETCS functions. The ERTMS/ETCS language (refers to SUBSET-026 chapter 7 and 8) is used for transmitting information over the radio, EuroBalise and loop air-gaps and the STM interface. It is based on variables, packets, messages and telegrams.

- **Establishing a communication session** OBU shall never have two different communication sessions established at the same time with the same RBC or with two different

- **Transitions Table** Most of the safety behavior is yielded by the model itself and it is difficult to produce a set of declarative properties, and to avoid paraphrase. In Isolation mode, OBU has no more responsibility and is isolated from the brakes, we can consider that the transition to or from this mode is safety relevant.

- **Procedure On-Sight** Fault Three Analysis

5.5 Tools Comparison

Table 5.1 and 5.2 show the tools that fulfills the CENELEC and the OpenETCS Criteria.

CENELEC EN50128 Criteria	ProR	Papyrus/ SymML	SCADE System	SCADE Suite	OpenModelica	Polarsys	MaTeLo	RT-Tester	Diversity
Formal Methods	X			X	X	X			
Defensive Approach		X	X	X		X			
Decision Table	X			X		X			
Modeling		X	X	X	X	X			
Structured methodology	X	X	X	X	X	X			
Design and coding standard			X	X					
Strongly typed programming language		X	X	X	X	X			
Fault detection & diagnostic			X	X		X	X	X	X
Failure assertion programming			X	X		X			
Diverse programming			X	X	X	X			
Memorizing executed cases			X	X	X	X	X	X	X
Effect analysis			X	X		X	X	X	X
Fully defined interface	X	X	X	X	X	X	X	X	X
Modelign supported by design		X	X	X	X	X			

Table 5.1: tools comparison with CENELEC criteria

OpenETCS Criteria	ProR	Papyrus/ SymML	SCADE System	SCADE Suite	OpenModelica	Polarsys	MaTeLo	RT-Tester	Diversity
State Machines		X	X	X	X	X			
Arithmetic and braking curves				X	X	X			
Truth table and logical statement				X	X	X			
Data Structure		X	X	X	X	X			
Establishing a communication Section			X	X	X	X			
Transition Table		X	X	X	X	X			
Procedures On-sight		X	X	X	X	X	X	X	X

Table 5.2: tools comparison with OpenETCS criteria

Conclusion and Future Work

The future EU Railway needs

Rail is a vital part of European Union transport, with a key role in addressing rising traffic demand, congestion, fuel security and decarbonization. But many European rail markets are currently facing stagnation or decline.

Faced with this reality, the Commission is proposing far reaching measures to encourage more innovation in EU railways by opening EU domestic passenger markets to competition, as well as accompanying technical and structural reforms.

These reforms regard three main changes:

1. **Promote Interoperability:** allow train to cross the border inside Europe and the movement between countries with a Technical, Regulatory and Operational constraints.

2. **More restrictions, standardizations and constraints into the development process**

3. **Improve the satisfaction of the customers**

In that way, OpenETCS project thinks to move forward in order to reduce life software and hardware cycle costs, use a formalization for certification (CENELEC) and improve the interoperability for seamless operation. For that reasons the project shall **reduce the complexity** using Standardizations, **reduce ambiguities** using formal approaches, **avoid "bug" surprise** with life-time service and **No vendor** lock-in using **Open proofs**.

Open Proofs is a new approach for safety and security critical systems and a development of the so called "Open Source Software". The aim of this *approach* is not just apply licensing concepts to the final software but also to the entire life cycle and all software components involved, including tools, documentation, verification maintenance and in particular safety case document.

Conclusions

At this point it is important to have a brief review on what are the results of this work, taking also into account the future needs, processes and methods.

- The CENELEC standard have its fault and weakness. There are contradictions that must be rectified, development methods and tools are changing and with them development live cycle which influences the entire standards.
- OpenETCS should have no-vendor lock-in, support migration strategy of the ERTMS, innovate concurrently with the European Union railway market.
- Promotion and acceleration of innovation development with an adequate safety, reliability and quality of service will reduce the lifecycle cost and enhance the security for software.
- Model Driven Engineering is a growing practice that can give a significant effort to the future development process with its adaptivity and flexibility but it needs to be more strictly standardized also to promote the interoperability between different tools and methods. It can guarantee also a very high cost reduction for the life-cycle.
- Agile Methods shall be modified and embedded specifically for the development of safety-critical system and according with the standards constraints and rules.
- At the moment tools open source such as ModelicaML and Polarsys can support just the non-safety process. To have a clear standardized code SCADE is highly recommended.

- An agreement between different tools of exchanged data in terms of types definition and format will fix all the heterogeneity problems. A toolchain at this point can be easily built and embed for the development process decisions.

OpenETCS and its follow up projects can contribute to find solutions and innovate this sector

Acknowledgements

Ringrazio innanzitutto il relatore di questa tesi, Prof. Stefano Russo per la grande disponibilità, l'aiuto costante, e la fiducia dimostrami. Desidero comunicargli tutta la mia gratitudine per avermi dato la possibilità di realizzare una fortemente desiderata esperienza di studio all'estero e per il prezioso supporto nella risoluzione di tutti i problemi che si sono presentati durante il lavoro.

Ringrazio i docenti del corso di laurea in Ingegneria informatica dell'Università Federico II di Napoli, per gli insegnamenti, non solo accademici, ricevuti in questi formativi anni di vita universitaria.

Un particolare ringraziamento lo rivolgo ai membri del gruppo OpenETCS, Peter, Baseliyos, Peymann, Bernd, Dr. Hase e ai numerosi partner del progetto, per l'ospitalità, la calorosa accoglienza e l'opportunità concessami.

Mi e doveroso ringraziare anche tutti gli amici che mi sono stati accanto in questo lungo percorso che con il loro supporto e la loro allegria mi hanno spesso distolto dalle preoccupazioni e dallo scoraggiamento.

Rivolgo estrema gratitudine alla mia famiglia, in particolare, sono profondamente debitore ai miei genitori, che hanno sempre assecondato le mie scelte, e con il loro incrollabile sostegno morale ed economico, mi hanno permesso di raggiungere questo traguardo.

Un grazie di cuore a Lara per aver creduto in me sin dal primo momento, per avermi dato conforto e sostegno, per essermi stata sempre accanto.

Giovanni

Bibliografia

[1] *IEC 62279: Railway applications – Communications, signaling and processing systems – Software for railway control and protection systems*. IEC CENELEC, March 2001.

[2] http://www.all4tec.net, June 2014.

[3] http://www.openetcs.org, June 2014.

[4] Paul Baker, Zhen Ru Dai, Jens Grabowski, Øystein Haugen, Ina Schieferdecker, and Clay Williams. *Model-driven testing*. Springer, 2008.

[5] Thomas Bardot. Evaluation model of ETCS using sysML and enterprise architect. Tool and model presentation, ITEA2 Project, March 2013.

[6] Marc Behrens, Hardi Hungar, Ana Cavalli, Jens Gerlach, Hansjörg Manz, Jan Welte, and Cyril Cornu. *openETCS Validation and Verification Strategy*. openETCS Validation and Verification Strategy Work Package, May 2013.

[7] Sami Beydeda, Matthias Book, Volker Gruhn, et al. *Model-driven software development*, volume 15. Springer, 2005.

[8] Gabriella Carrozza, Mauro Faella, Francesco Fucci, Roberto Pietrantuono, and Stefano Russo. Integrating mdt in an industrial process in the air traffic control domain. In *Software Reliability Engineering Workshops (ISSREW), 2012 IEEE 23rd International Symposium on*, pages 225–230. IEEE, 2012.

[9] Gabriella Carrozza, Mauro Faella, Francesco Fucci, Roberto Pietrantuono, and Stefano Russo. Engineering air traffic control systems with a model-driven approach. *Software, IEEE*, 30(3):42–48, 2013.

[10] EN CENELEC. 50128. *Railway Applications: Software for Railway Control and Protection Systems, European Committee for Electrotechnical Standardization (CENELEC)*, 1997.

[11] Guillaume Pottier David Mentre, Stanislas Pinte and WP2 participants. *Methods and tools benchmarking methodology*. ITEA 2 Project, March 2012.

[12] Mourad Debbabi, Fawzi Hassaine, Yosr Jarraya, Andrei Soeanu, and Luay Alawneh. *Verification and Validation in Systems Engineering*. Springer, 2010.

[13] Lenny Delligatti. *SysML Distilled: A Brief Guide to the Systems Modeling Language*. Pearson Education, 2013.

[14] Francesco Flammini and IGI Global. *Railway safety, reliability, and security: Technologies and systems engineering*. Information Science Reference, 2012.

[15] Sanford Friedenthal, Alan Moore, and Rick Steiner. *A practical guide to SysML: the systems modeling language*. Elsevier, 2011.

[16] Peter Fritzson. *Principles of object-oriented modeling and simulation with Modelica 2.1*. John Wiley & Sons, 2010.

[17] Object Management Group. *Requirements Interchange Format (ReqIF)*. Object Management Group, 1.1 edition, October 2013.

[18] Object Management Group. *Object Management Group Model Driven Architecture (MDA)*. OMG, june 2014.

[19] Klaus-Rüdiger Hase. "open proof" for railway safety software-a potential way-out of vendor lock-in advancing to standardization, transparency, and software security. In *FORMS/FORMAT 2010*, pages 5–38. Springer, 2011.

[20] Elizabeth Hull, Ken Jackson, and Jeremy Dick. *Requirements engineering*, volume 3. Springer, 2005.

[21] Jan Jürjens. Developing safety-critical systems with uml. In *«UML» 2003-The Unified Modeling Language. Modeling Languages and Applications*, pages 360–372. Springer, 2003.

[22] Barbara Ann Kitchenham. Evaluating software engineering methods and tool part 1: The evaluation context and evaluation methods. *ACM SIGSOFT Software Engineering Notes*, 21(1):11–14, 1996.

[23] John C Knight. Safety critical systems: challenges and directions. In *Software Engineering, 2002. ICSE 2002. Proceedings of the 24rd International Conference on*, pages 547–550. IEEE, 2002.

[24] O Levêque and P De Cicco. ETCS implementation handbook. *Infrastructure Department, UIC*, 2008.

[25] Marielle Petit-Doche Micheal Jastram and all partecipants of the decision process. Report on the final choice of the primary toolchain. ITEA 2, October 2013.

[26] Jos Vrancken Michel dos Santos Soares. Model-driven user requirements specification using sysml. *JOURNAL OF SOFTWARE*, 3(6):57, June 2008.

[27] MOF OMG and QVT Final Adopted Specification. Object management group. *Retrieved from*, 2007.

[28] Mahlmann Peter, Jacob Baseliyos, and Rieger Stefan. Initial exploitation plan and standardisation strategy, open source business model. OpenETCS, Volkenstrasse 5, Munich, December 2013.

[29] Stanley Peter. *ETCS for Engineers*. Eurail press, 2011.

[30] S Pressman Roger. Software engineering: a practitioner's approach. *McGrow-Hill International Edition*, 2005.

[31] Rail Safety and Standars Board Limited. ETCS system description. RSSB Block 2, Angel Square 1 Torrens Street London EC1V 1NY, 2010.

[32] Wladimir Schamai. Modelicaml: Getting started. Technical Report 1.3, EADS Innovation Works - Linkoping University, May 2010.

[33] Wladimir Schamai, Peter Fritzson, Chris Paredis, and Adrian Pop. Towards unified system modeling and simulation with modelicaml: modeling of executable behavior using graphical notations. In *Proceedings 7th Modelica Conference, Como, Italy*, 2009.

[34] Ken Schwaber and Jeff Sutherland. The definitive guide to scrum: The rules of the game, july 2013.

[35] Bran Selic. The pragmatics of model-driven development. *IEEE software*, 20(5):19–25, 2003.

[36] Richard Soley and the OMG Staff Strategy Group. Model driven architecture. *White Paper - Object Management Group*, November 27, 2000.

[37] T Stålhane, T Myklebust, and GK Hanssen. The application of safe scrum to iec 61508 certifiable software. ESREL, 2012.

[38] Tor Stålhane-IDI. Safety standards and scrum–a synopsis of three standards.

[39] Jeff Sutherland. Scrum handbook. *Online (12.10. 2012): jeffsutherland. com/Scrumhandbook. pdf*, 2010.

[40] OMG SysML. Omg systems modeling language, 2006.

[41] Hirotaka Takeuchi and Ikujiro Nonaka. The new new product development game. *Harvard business review*, 64(1):137–146, 1986.

[42] SRS UNISIG. ERTMS/ETCS - class 1 subset 026. *Issue*, 2(0):22, 1999.

[43] Mark Utting and Bruno Legeard. *Practical model-based testing: a tools approach*. Morgan Kaufmann, 2010.

[44] Tim Weilkiens. *Systems engineering with SysML/UML: modeling, analysis, design*. Morgan Kaufmann, 2011.

[45] Doug Rosenberg with San Mancarella. *Embedded System Development using SysML*. Sparx Systems Pty Ltd and ICONIX, 2010.