 THE RESEARCH LEADING TO THESE RESULTS HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S SEVENTH FRAMEWORK PROGRAM (FP7/2007-2013) FOR CRYSTAL – CRITICAL SYSTEM ENGINEERING ACCELERATION JOINT UNDERTAKING UNDER GRANT AGREEMENT N° 332830 AND FROM SPECIFIC NATIONAL PROGRAMS AND / OR FUNDING AUTHORITIES.

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

Use Case Requirements Specifications
D501.020

DOCUMENT INFORMATION

Project	CRYSTAL
Grant Agreement No.	ARTEMIS-2012-1-332830
Deliverable Title	Use Case Requirements Specifications
Deliverable No.	D501.020
Dissemination Level	CO
Nature	R
Document Version	V1-0
Date	2014-02-07
Contact	Renato De Guglielmo
Organization	Ansaldo STS (ASTS)
Phone	+39 (0)81 243 7608
E-Mail	Renato.DeGugliemo@ansaldo-sts.com

Authors

Name	Company	E-Mail
DE GUGLIELMO Renato	Ansaldo STS	Renato.DeGugliemo@ansaldo-sts.com
VELARDI Luigi	Ansaldo STS	Luigi.Velardi@ansaldo-sts.com
NARDONE Roberto	Ansaldo STS	Roberto.Nardone.Prof608@ansaldo-sts.com
MASSAROLI Gianpaolo	Ansaldo STS	Gianpaolo.Massaroli.Prof644@ansaldo-sts.com
MARRONE Stefano	Second University of Naples	stefano.marrone@unina2.it
VITTORINI Valeria	University of Naples "Federico II"	valeria.vittorini@unina.it
GENTILE Ugo	University of Naples "Federico II"	ugo.gentile@unina.it

Reviewers

Name	Company	E-Mail
BARBERIO Gregorio	Mate Consulting	g.barberio@mateconsulting.it
SCHERRER Christoph	Thales AT	christoph.scherrer@thalesgroup.com
VALLÉE Frédérique	All4Tec	frederique.vallee@all4tec.net

Change History

Version	Date	Reason for Change	Pages Affected
0-1	10/01/2014	Create document	All
0-2	24/01/2014	Introduction of internal review comments and updates	All
1-0	07/02/2014	Introduction of external review comments and updates	All

CONTENT

Content of Figures

Content of Tables

1 Introduction

1.1 Role of Deliverable

This document has the following main purposes:

- defining the details of ASTS use case.
- collecting all the requirements specifications for bricks to integrate to be adopted in ASTS use case.

1.2 Relationship with Other CRYSTAL Documents

This document is strictly connected with the deliverable “CRYSTAL_D_D501.010 – Data and Methodologies report”, because the system workflow presented here is completely based on the modelling methodology chosen to be adopted in ASTS use case and described there. Then, another link between the two deliverables is represented by the translation of the main methodological requirements (presented in D501.010) into the technological requirements necessary for the ASTS needs (presented here).

This document is also connected with the deliverable “CRYSTAL_D_D603.011 – Specification, Development and Assessment for System Analysis and Exploration”, where the background and the state of the art on the adopted technologies are reported.

Furthermore, this document is also linked with the deliverable “CRYSTAL_D_ D612.011 – Specification, Development and Assessment for Validation Models - V1”, in particular with its Sections 10, 11 and 12, because in these sections the bricks (and their Technical Items) which should satisfy all the technological ASTS needs are listed and described.

1.3 Structure of This Document

This document is structured as follows:

- Section 1 (this Section) introduces the contents and the structure of the document, clarifying also the relationships with other documents related to the CRYSTAL project;
- Section 2 provides a description of ASTS use case: at the beginning, it will focus on the problems which have led Ansaldo STS to join the Crystal project; then on the desirable solutions (i. e., the goals to be achieved); and finally on the application example in which these solutions are applied;
- Section 3 describes how the validation approach adopted in ASTS use case is implemented in the system workflow;
- Section 4 lists, starting from the main methodological requirements, all the requirements specifications for bricks to integrate to be adopted in ASTS use case;
- Section 5 reports the list of acronyms used in this document;
- Section 6 reports the list of references.

2 ASTS Use Case: the Radio Block Centre

2.1 Use Case Context

The increasing complexity of railway systems requires an evolution of the validation approach in order to be more effective with minor time and costs efforts. These activities shall be conducted by the V&V team which is independent from the development team and which should not know any information about the development (mandatory since CENELEC standards are applicable). This team shall rely only on the information contained in the requirements and in high-level behavioral description. Given the high complexity, the actual control systems require the definition and the executions of thousands test cases, which shall be able to verify also parallel execution flows. A great effort is spent not only in the definition phase, but also in the realization of executable test scripts and in the log analysis. An improvement of the actual validation approach is hence required in the rail domain with the aim to automatize standard and repetitive operations and to introduce modern technologies in these activities.

One of the most critical components installed in modern railways is the signalling system which aims at guaranteeing the complete control of the railway traffic, with a high-level of safety, essentially to prevent trains from colliding. Actual signalling systems implements complex protocols to exchange information with the trains and their implementation requires the usage of a high number of hardware components which execute complex software. These systems shall be validated against system requirements, given by the client and by applicable standards and norms ([CENELEC 50126, 2012], [CENELEC 50128, 2011]).

Since the signalling system is one of the most complex and critical, the use case chosen by ASTS in the CRYSTAL project is the Radio Block Centre (RBC) system which represents the main component of a possible implementation of European Rail Traffic Management System / European Train Control System (ERTMS/ETCS) (for further details refer to Section 2.3). The RBC system collects all the information about the positioning of trains and communicates with them giving, between other, movement authorization along track portions.

The validation approach, actually implemented in ASTS, is centred on the system requirements, reported in appropriate documents. With reference to the RBC system, these requirements are collected in the “RBC System Requirements” document which contains also a behavioural description of the entire set of functions implemented by the RBC (e.g. starting of mission procedures, ending of mission procedures, movement authority granting, etc.). This document, hence, acts as input for the implementation and the validation activities.

In the first step V&V Engineers try to formalize the system behaviour with the help of state-based formalisms (highly recommended by the standards during these phases of the applicable lifecycle). Then requirements are analysed and Test Specifications, able to demonstrate the validity of the requirements on the implemented system are defined. In a following phase appropriate Test Cases are obtained from the set of Test Specifications, which are then translated into executable tests (the RBC Test Scripts). RBC Test Scripts are executed on the implemented system, both in simulated and in real environments. Each test execution is logged on complete and complex Test Log files, which are then analysed in order to verify the correctness referring to the expected outcomes. The test outcomes, in a more essential and comprehensible format, are also reported in the “RBC Test Report” document, where also the mapping on the initial requirements is highlighted.

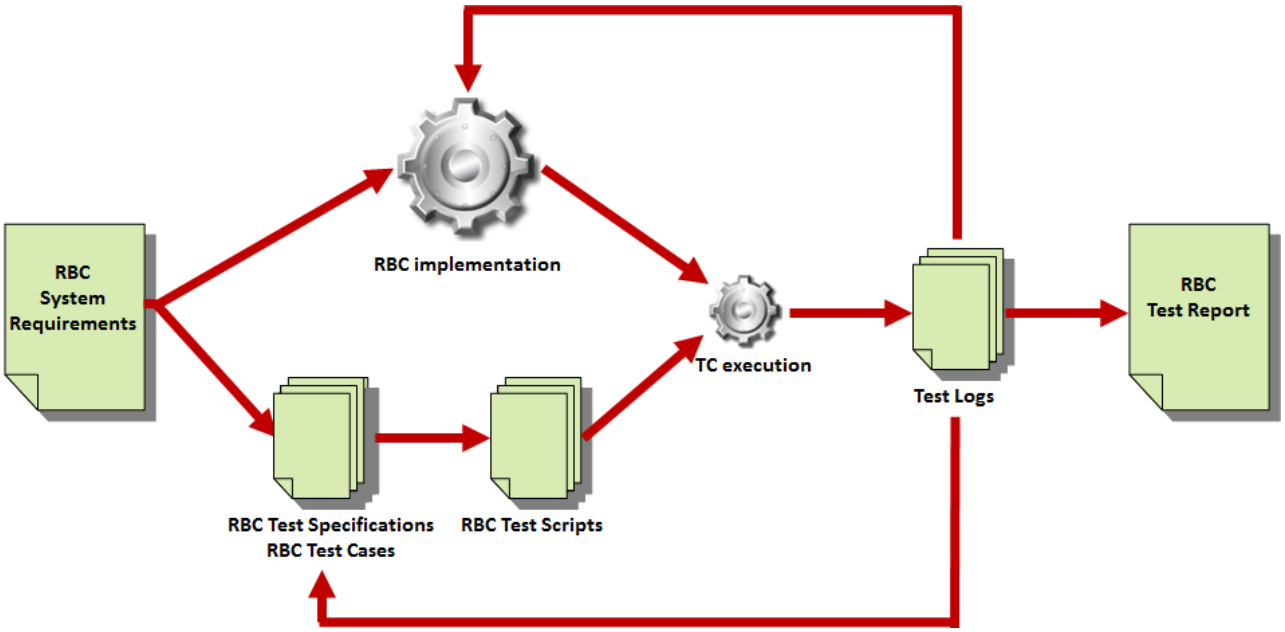

Figure 2-1: ASTS validation approach

Figure 2-1 depicts the validation approach implemented by ASTS as previously described. The two backward arrows indicate the feedbacks of the test execution on the implementations and/or on the Test Specification (if some tests individuate bugs) of the RBC. These two arrows indicate that, if the outcomes of a test are not those expected, a bug can be present in the implementation of the system and/or an error has been made during the system modelling performed by the V&V Engineers.

In the actual validation approach, ASTS spends a high effort (in terms of time and costs) during the definition of Test Specifications and Test Cases. This activity is manually performed by experts through the definition of a system behavioral model. Test Cases are written in executable and proprietary language in order to be executed on the real systems or on simulators. Execution logs are also manually analyzed. Manual activities can also introduce errors. Furthermore changes of the requirements imply the manual identification of the tests impacted by these changes and their update. Moreover, the model used to define the tests by V&V team is only a representation of the system behavior and usually doesn't allow any automatic verification of its feasibility, as well as it's not possible to define, automatically, any system test from the model itself.

Another problem which arises during the validation of a complex railway system is represented by the fact that, in order to guarantee the interoperability among several sub-systems provided by different suppliers, general test scenarios involving different sub-systems have to be defined and executed on them, but, actually, each company has its own language to "feed" the system under test, and hence testing the interoperability requires that each supplier adapts the general test scenarios to its own language, in order to be able to execute them on its own environment.

2.2 Use Case Goals

For the above reasons (Section 2.1), having the opportunity to use a new methodology thanks to which the model becomes a sort of “natural step” in the test definition process (so that, once the model is defined, the test cases can be semi-automatically generated from it), would imply a significant reduction of time and costs during the validation phase, limiting considerably the effort spent by V&V team.

Indeed, the traceability of the model, both on system requirements and on generated tests, could support the engineers in the analysis of the impact of modifications in system requirements during the whole life cycle of the system, reducing time needed to identify the impacted tests and to modify them after changes in requirements.

Furthermore, the automatic traceability between requirements and tests would simplify the maintenance of entire test suite and the analysis of the results, speeding up the identification of requirements or parts of the system not rightly implemented.

Another desirable improvement during the validation phase of a complex railway system could consist in realizing a modular structure of the tool chain, with the adoption of a standard language: in fact, it would allow different companies/suppliers to share all the steps through the definition of general test scenarios, reducing the risk of misunderstanding/incoherence and facilitating the setup of a multi-company interoperable testing environment, speeding up the assessment of the overall system in different countries.

Moreover, by using a common language to define tests, the execution of interoperability tests could be performed in laboratory and not by means of practical feasibility, with a considerable cost reduction.

2.3 Use Case Description

The use case chosen by ASTS, in which the new methodology provided by CRYSTAL project is applied, is the RBC (Radio Block Centre) system, the main component of Level 2 of European Rail Traffic Management System / European Train Control System (ERTMS/ETCS).

The ERTMS is an initiative backed by the European Union to enhance cross-border interoperability and the procurement of signalling equipment by creating a single Europe-wide standard for train control and command systems. The main component of ERTMS is the ETCS, a signalling, control and train protection system designed to replace the many incompatible safety systems currently used by European railways, especially on high-speed lines: it allows a train equipped with ERTMS/ETCS to travel without signal system boundaries within the ERTMS/ETCS fitted infrastructure network, regardless of the country the train is travelling in, the legal nature of the infrastructure manager or the supplier providing the ERTMS/ETCS system.

The ERTMS concept has been developed in 4 different functional levels, depending on the system architecture. Level 0, level 1 and level 2 of ERTMS/ETCS are already implemented (while Level 3 is currently under development) and in revenue service in most European countries and beyond, with Ansaldo STS acts as global leader in ERTMS/ETCS components and systems for Conventional and High-Speed lines.

The definition of the ERTMS level depends on how the route is equipped and the way in which the information is transmitted to the train:

- it's possible to talk of ERTMS/ETCS Level 0 when ERTMS/ETCS-compliant locomotives or rolling stock interact with line-side equipment that is non-ERTMS/ETCS compliant. A driver shall observe the physical signals encountered during the track, knowing the specific meaning of those signals on the railway;
- ERTMS/ETCS Level 1, instead, consists of a cab signalling system that can be superimposed to the existing conventional signalling system leaving the fixed signal system (national signalling and track-release system) in place. The on-board equipment monitors and calculates the maximum speed and the braking curve relying on the data received from the balises at fixed point;
- ERTMS/ETCS Level 2 is a train protection system based on continuous communication of variable data between the RBC and the trains via a radio system (some additional information is received on board via fixed balises);
- finally, ERTMS/ETCS Level 3 implements a full radio based train control and spacing hence fixed track equipment is no longer required. This technology allows for detecting the current position of each train always in time, hence it is possible to send continuously line-clear authorization to each train.

As described in Figure 2-2, in ERTMS/ETCS Level 2 the RBC monitors continuously the train movements because it automatically finds out from trains their exact positions, receives train detection and route status information from the Interlocking and Automatic Block System as applicable, making all information available to each train continuously via GSM-R in form of movement authorities. The RBC interacts with the on-board by managing a Communication Session using the EURORADIO protocol and the GSM-R network. A single RBC can manage contemporary until a fixed maximum number of trains depending on physical characteristics of the GSM-R network.

Different RBCs are implemented for different railway projects. Obviously all the validation activities shall be afresh performed but the implementation of a new RBC can reuse some of the activities and of the artefacts produced for a different version. For this reason RBC is commonly structured in a portion common to different implementation (the **RBC core**) and a portion which is coherently updated in each project (the **RBC specific**). Requirements and test cases related to the RBC core between different projects are commonly the same, hence the V&V activities should benefit from this situation in order to reduce time and costs. On the other hand, the detection of a bug on this portion a real implementation should generate an alert which requires a further verification on the other implementations.

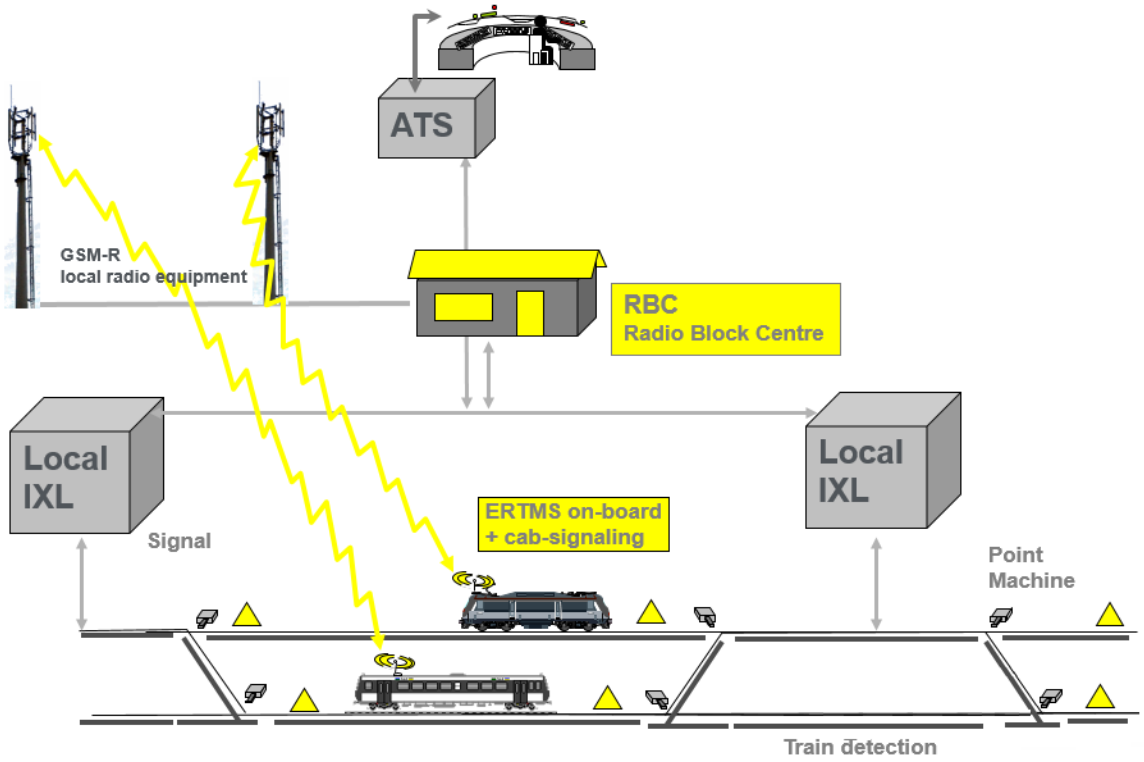

Figure 2-2: ERTMS/ETCS – Level 2

3 The Crystal workflow

3.1 General description

The Crystal methodology, deeply described in the D501.010, shall improve the effectiveness (in terms of time and costs) of the validation approach actually performed by ASTS. In synthesis a set of test scripts shall be generated and executed on the real systems (in simulated or real environments) in order to demonstrate the validity of the system requirements.

The validation of a new signalling system requires the implementation of a complex workflow, which starts with the requirements analysis and ends (when no further corrective actions are required) with the generation of appropriate Test Reports where the fulfilment of each requirement is demonstrated through the validity of a test set. The detailed workflow is depicted in Figure 3-1, where all the necessary steps to validate a system are depicted.

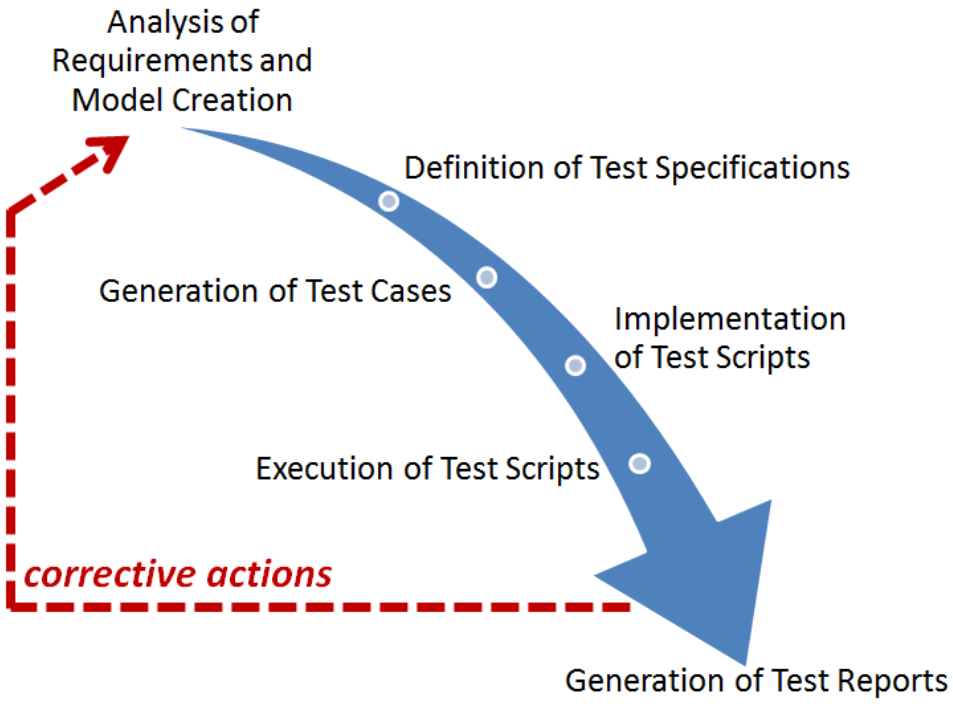

Figure 3-1: The Crystal workflow

The entire workflow starts from the “Analysis of Requirements and Model Creation” step. In this phase a model is created on the basis of the information contained in the system requirements. The generation of this model during validation phase allows conducting also a consistency and completeness check: if some requirements have not been used during the modelling phase, it means that this requirement is unnecessary or redundant. Otherwise a set of requirements could lead to conflicting portion of the model: in this case these requirements need to be verified and interpreted in order to produce an unambiguous model of the system behaviour.

After the creation of the model, the “Definition of Test Specification” step is performed. In this phase each requirement is formalized as the system model and leads to one or more Test Specifications. A Test Specification, hence, verifies one or more requirements: this link shall be clearly traced.

The workflow goes on with the “Generation of Test Cases”. In this phase Test Specifications are properly translated into Test Cases which express all the steps of a test in terms of necessary inputs and expected outputs. This step shall be automatically performed by proper methodologies and tools. The Test Cases are, in this phase, written in a formal language comprehensible by the V&V Engineers. It is necessary, in fact, to give them the possibility to operate on Test Cases and to have an idea of the model portion stressed by each Test Case; hence each step of the Test Case shall also clearly identify states/transitions to which it is linked. Test Cases could be also manually updated but it is not recommended.

When the set of Test Cases is clarified, the “Implementation of Test Scripts” step is performed. In this phase Test Case, reported in a comprehensible language, are properly translated into executable scripts in order to be executed on the simulated/real system. Since different systems are involved, it is preferable that these tests are written in a common language in order to be executed by different simulators of different industries.

Test Cases are then executed on the real or on simulated environments. This step is taken in charge by each railway operator in a different way since it depends of the specific implementation on the systems and of the simulation environments. For this reason, the improvement of this step is external to the Crystal project. The execution of Test Cases produces Test Logs where the execution is logged to verify the correctness with respect to what expected.

Finally Test Logs are inputs for the “Generation of Test Reports” activity. In this step Test Logs are automatically processed in order to produce Test Reports where it is reported the outcome of the test and the not satisfied requirements are annotated.

In the following paragraphs the adoptable technologies are listed for each step of the Crystal workflow. A great attention is given to the technologies which enable the automatic generation of Test Cases, since the improvement of this step is the most complex from a methodological and technological point of view.

3.2 From System Requirements to Test Cases

According to the model driven test case generation process, described in the deliverable “CRYSTAL_D_D501.010 – Data and Methodologies report”, a Domain Specific Modeling Language (DSML) will be developed and used to model the behavior of the system under test (SUT) and the test specifications. In the same deliverable some requirements have been specified for this process, resulting is a Finite State Machine formalism provided with a formal semantics and specific features in order to be applied to UC5.1. Hence, some steps have to be accomplished to define the DSML in the WP6.12. The first of them is the definition of a proper syntax. At this aim a meta-modeling language must be chosen. A meta-modeling languages in Model Driven Engineering (see the background in “CRYSTAL_D_D603.011 – Specification, Development and Assessment for System Analysis and Exploration”) defines the abstract syntax of a DSML, generally in form of a class diagram (called meta-model) [Harel, 2004]. Meta-models capture domain specific concepts and their relationships (semantic mapping). Then, the abstract syntax is mapped to a concrete

syntax, i.e. the DSML constructs (syntactic mapping) [Harel, 2004], [Clark, 2001]. The relationships between abstract syntax, concrete syntax and semantic domain are illustrated in Figure 3-2.

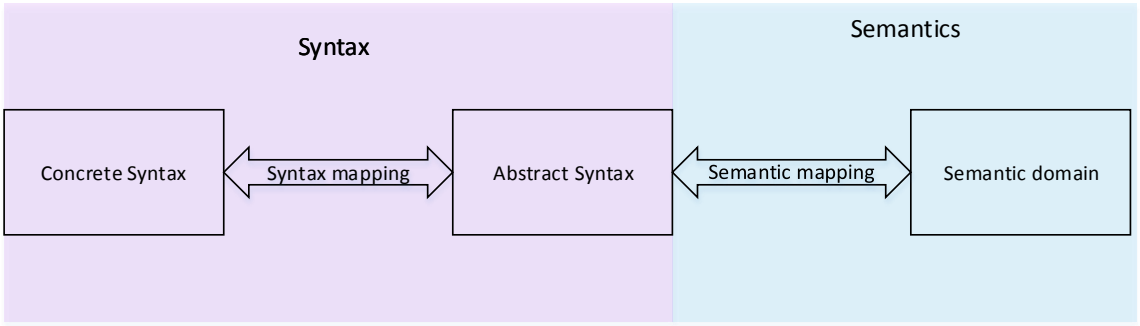

Figure 3-2: DSML definition: semantic and syntactic mappings

The relationships among meta-model, modelling language and model are illustrated in Figure 3-3.

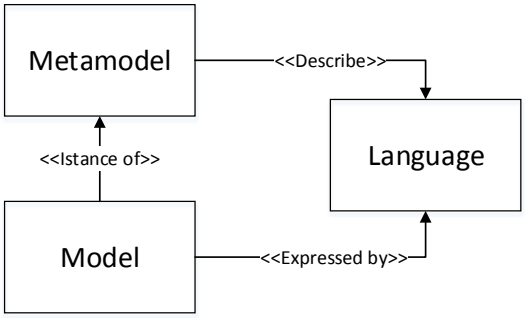

Figure 3-3: Meta-model, Model, Language

Hence, the choice of the meta-modeling language is of great importance, as this choice has consequences on the entire process of test generation. In the next Subsection we describe the main alternatives and how they determine different ways to implement and support the generation process.

3.2.1 Meta-modeling solutions

In evaluating different solutions, we have taken into account three main coordinates, i.e. if the meta-modeling language enables:

1. the usage of easy-to-understand and user-friendly environments/workbenches to develop the models (Editing);
2. the possibility of performing static analysis of the model during its construction (Analysis);
3. the possibility of executing the models, i.e. of performing simulations (Simulation).

According to these coordinates, we have grouped the different solutions into four categories:

4. Meta Object Facility (MOF) based: MOF is a OMG standard and is the basis of MDA (Model Driven Architecture);

5. Eclipse Modelling Framework (EMF) based: this solution exploits the power of Eclipse, it might also be conjugated with the usage of UML;
6. Grammar based: the meta-model is defined by using grammars like BNF or EBNF;
7. Other approaches: several approaches belong to this category, both open source and commercial solutions. As example, MetaEdit+ is a commercial suite that enables the generation of full code directly from models and allows for defining a new domain specific language from scratch.

Figure 3-4 provides a bird-eye view of the different possibilities and consequences on Editing, Analysis and Simulation. Here below, they are described in more details.

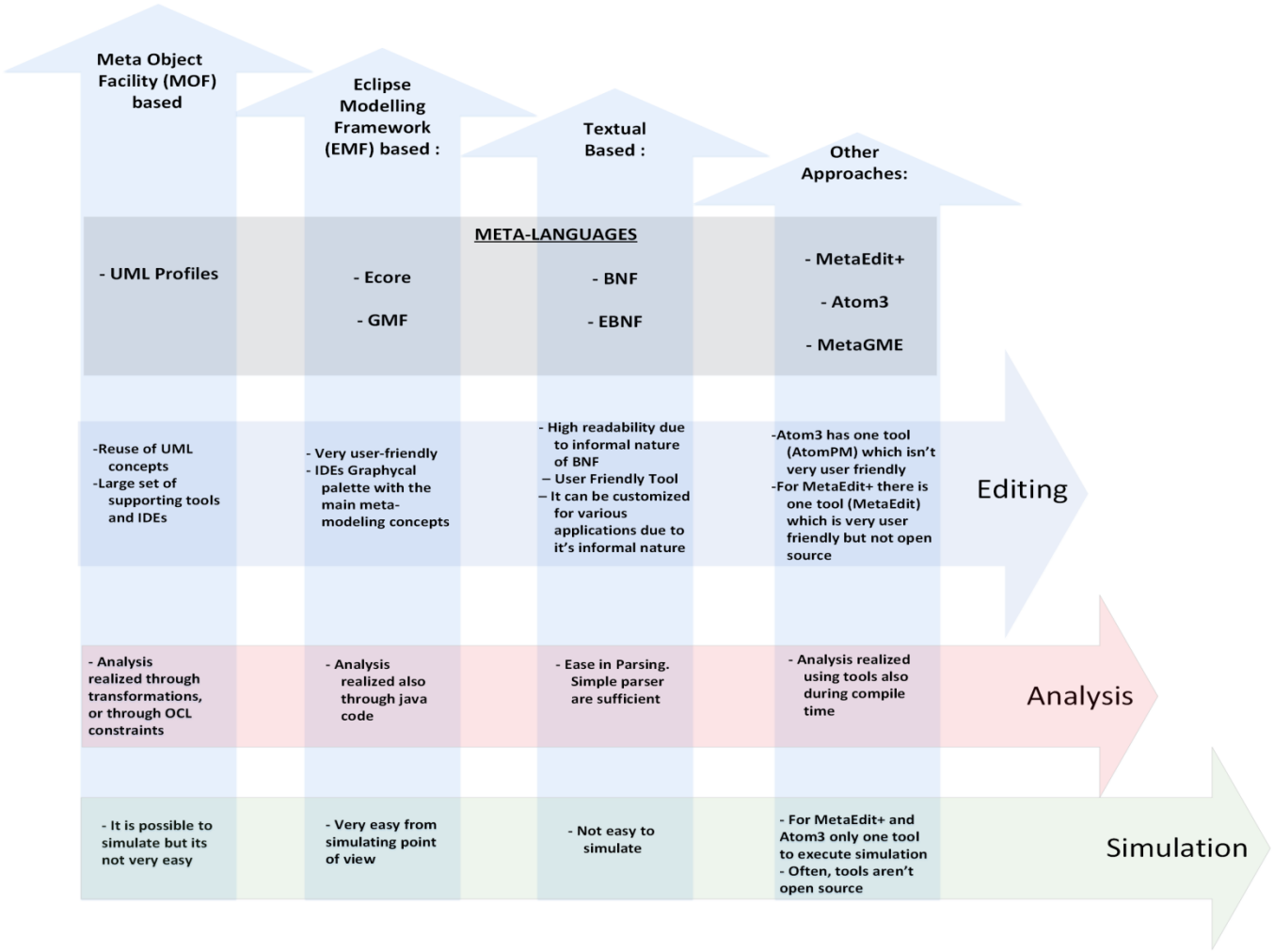

Figure 3-4: Meta-modelling solutions

3.2.1.1 MOF- Meta Object Facility

Meta Object Facility (MOF) is a standard produced by OMG which is place at the top layer of Model Driven Architecture.

MDA is a consolidate approach in which models and modelling techniques are the main artefacts of software development cycle. The MDA architecture consists of four layers:

- M3, the meta-meta-model layer, which provides a model of the modelling language;
- M2, the meta-model layer, which describes the concepts used by the modelling language to construct the model within M1 layer;
- M1, the model layer, in which there are the models of the element of the system. Layer M1 provides the generalization of concepts in M0 layer;
- M0, the system layer.

MOF provides a language to describe modelling language hence it is defined at the M3 layer. Within MOF it is possible to create both a new language and extensions of existing language.

This second approach is the one on which the UML profiles are based. An UML Profile allows for extending UML models for specific domains and platforms. The extension could be made by using:

- Stereotypes, used to extend UML concepts, so providing constructs to build the models. Graphically the application of a Stereotype is identified by the label <<Stereotype>> before its name on a UML element; it is also possible to define icons which are showed after the stereotype application;
- Constraints, associated to stereotypes, which are used to impose restrictions to the stereotype. OMG has defined a language named OCL to express a constraint. Any rule associated to the stereotype can be expressed by using a constraint;
- Tagged values, which are meta-attributes associated to stereotypes or meta-classes. Each tagged value has a name and a type associated to a specific stereotype.

Several UML Profiles are available; some of them are defined or standardized by OMG itself. Examples of UML profiles from OMG are MARTE (Modeling and Analysis of Real-Time and Embedded systems) and UTP (UML Testing Profile). UTP provides extensions to UML in order to support the design, visualization, specification, analysis, construction, and documentation of the artefacts involved in testing activities [Dai, 2004].

To understand the reasons behind the adoption of the UML Profile we show an example depicted in Figure 3-5. In this example the stereotype “Figure” is added to UML meta-model by extending the “Class” meta-class. The properties of the figure, i.e. colour and geometry, are described by using homonymic tagged value.

This example shows some relevant advantages of the UML Profiles. Firstly they allow to introduce specific concepts of an application domain at meta-model level (UML in fact does not have the concept of “figure”); secondly they allow to add expressive power at modelling language. UML profiles also provide additional information that can be used for M2M or M2T transformation [Fuentes, 2004].

Furthermore UML is widespread used during all software engineering processes; it is implemented by a large amount of tools and software development environments. Finally the interoperability is guaranteed by the XMI format, defined by the OMG group that allow the interchange of the same models between different tools.

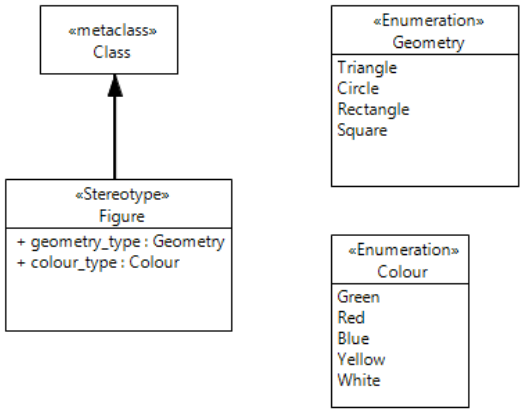

Figure 3-5: UML Profile: an example

3.2.1.2 EMF – Eclipse Modelling Framework

Eclipse Modelling Framework is a stable framework which provides facilities for building toolsets and Java applications based on model manipulation. EMF allows for creating a model and generating code from it with the same level of usability of an UML model.

As stated in previous Subsection, a model is an abstract representation of an object. A model can be described using several languages. EMF introduces a new concept which is the passage between different high-level representations. More specifically, as depicted in Figure 3-6, EMF unifies three relevant technologies: UML, XML and JAVA. As an example, EMF allows to transform a XML schema into a UML class diagram or directly in executable Java code [Steinberg, 2009]. The EMF is supported by the Ecore format that is a simple format guaranteeing interoperability between tools. The Ecore format is also an open format, easy to understand and manipulate, also in textual way.

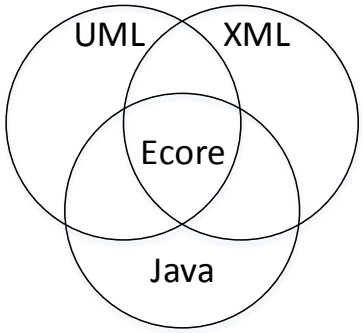

Figure 3-6: Ecore Technologies

Hence EMF integrates both modelling and programming principles. There are several advantages behind EMF-based solutions:

1. it is easy to provide a clear representation of what the system it is supposed to do;
2. Code generated from the model is high readable;

3. EMF adopts the Ecore format to provide interoperability and information interchange;
4. It is possible to perform queries on the structure of the model;
5. It is possible to generate an editor for a model with functionalities based on the model itself.

EMF consists of three main parts:

a. EMF.Core that includes a meta-model in Ecore format for describing models. Ecore is well supported by large set of API for manipulating EMF objects;
b. EMF.edit, that includes generic classes for building editors for EMF models. This framework allows to display EMF model in a standard Eclipse view and provides to manipulate models properties, classes etc.
c. EMF.codegen, which provides the code generation facilities. EMF.codegen is based on Java Development Tooling to build EMF editor. With EMF.codegen is possible to generate:
    a. Classes from model, including factory methods and packages
    b. Adapters that allow editing and display of generated classes;
    c. Editor, which allows to customize the model. The editor is show in eclipse-like view.

The development workflow of EMF is very simple: first a model is created and defined using the Ecore format. Second, when the model is defined, it is possible to generate java code from it.

Ecore includes few and essential concepts such those included in the Essential MOF (i.e. the meta-language at the basis of UML): some of the basic types included in it are Eclass, Eattribute, and Ereference which have the same meaning of the Class, Property and Reference of the Essential MOF. As an example, considering the same objects of the UML Profile described before, it is possible to have the Ecore diagram described in Figure 3-7. This diagram represents the meta-model describing the same domain reported in Figure 3-5; the difference is that the UML profile of Figure 3-5 adds the “figure” concept to UML while the Ecore of Figure 3-7 reports a new modelling language created from scratch: the usage of this domain model does not allow to use other concepts than the one of “figure”.

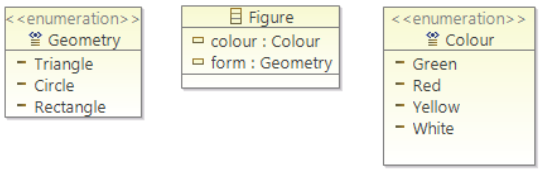

Figure 3-7: Ecore example

Here the concept of stereotype is replaced by the Eclass concept while tagged values are replaced by the Eattribute. Constraint can be inserted within Ecore diagram by using natural language or structured language like OCL.

3.2.2 Model Transformations

A key concept within MDE context is the transformation of models.

Transformations allow for obtaining a model from another in automatic way. The OMG's MDE standards specify the need for change to move from platform-independent models to platform-specific models, raising the level of abstraction during the modelling phase, and then reducing it for a specific platform, during the development stages. Model transformations may be useful, for example, to generate a formal model starting from a UML model.

Model transformations can be grouped into two categories:

- Model-to-Model (M2M) transformations: M2M transformations aim at transforming source models into other models, also expressed in different formalisms. The main motivation of their need in our context is that the new model enables to perform analyses that are not feasible in the previous formalism. Hence M2M transformations implement these mapping by defining proper sets of rules between the source and the target languages. An example of language used to write M2M and M2T transformation is the ATLAS Transformation Language (ATL) [Jouault, 2006].
- Model-to-Text (M2T) transformations: M2Ts are able to generate text directly from a model (conformant to a specific meta-model). M2Ts have a paramount importance in model driven software development processes since automatic code generation represent a final but a necessary step in such processes. In a wider perspective, M2Ts can be used to generate text, reports, configuration files or to instantiate abstract models according to a specific concrete syntax. This last case can be used when a formal model, expressed for example by means of an Ecore based language, must be translated into a specific data format understandable by existing solvers. M2Ts can be divided into two categories according to the constituting principles as indicate in WP6.03.

Figure 3-8 summarizes the possible choices of model transformation technologies given the source meta-modelling language. An arc between a meta-modeling solution group and a language X (Java, ATL, QVT, etc.) means that a model transformation from/to the specified meta-modeling languages may be implemented by X.

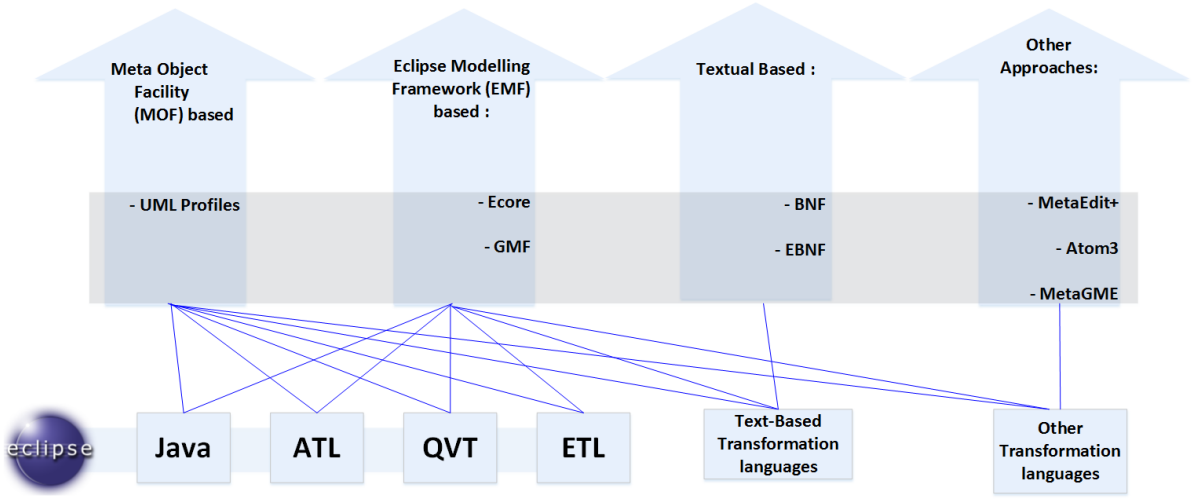

Figure 3-8: Transformation technologies

3.2.2.1 Transformation languages

Java (general purpose language)

Java is one of the most important programming languages based on object orient paradigm. Java may be also used to implement transformations between source and target languages. Despite the power of Java, specific transformation languages are preferred because they provide a range of useful features which facilitate writing, understanding and executing of the transformation.

ATL

ATL stands for Atlas Transformation Language, created by the ATLAS INRIA & LINA research group.

It is the answer to the OMG MOF and to QVT. It is a model transformation language specified both as a meta-model and as a textual concrete syntax. It is a hybrid language since it is possible to define declarative and imperative statements.

Most of the rules written by using ATL are declarative, which means that mappings can be expressed simply. Despite the declarative way is preferred, imperative constructs are provided in order to manage situations too complex to be also dealt by means of declarative rules.

An ATL transformation program is composed of rules that define how source model elements are matched and navigated to create and initialize the elements of the target models.

The structure of an ATL program is composed by four parts: Header, Import, Rules and Helpers. The header contains the transformation name and the declaration of both source and target model. The import section is used to import definition specified by other ATL modules. This can be done by using the keyword “uses” followed by the library name. Rules section is the core of ATL file because it contains the transformation rules. Each rule defines source patterns, the element type of the source model to be transformed, and the target pattern, used to generate a portion of the target model.

ATL supports the definition of helpers: a helper is used to declare functions and global variables used by the transformation rules. Helper functions are written by using the OCL language.

QVT

QVT stands for Query/View/Transformation and it is a language for model transformation created and standardized by OMG. Due to OMG derivation, QVT includes both MOF 2.0 and OCL 2.0 specifications. As ATL previously described, this language is hybrid (declarative and imperative).

With QVT are provided three different transformation languages:

a. QVT-Core which is a declarative language designed to be simple and to be used on QVT Relations target model. Due to the not fully specified nature of QVT-Core, QVT-Relational is more expressive.
b. QVT-Operational which is the imperative transformation defined for writing unidirectional transformations;
c. QVT-Relations, which provide declarative transformations. The transformation written by using QVT-Relational can be both unidirectional and bidirectional.

Compared to ATL QVT languages do not permit M2T transformations, since each model must conform to MOF 2.0 meta-model. M2T transformations are being standardised separately by OMG in MOFM2T standard [MOFM2T, 2008].

ETL

ETL stands for Eclipse Transformation Languages and provides M2M transformation language features to Epsilon. ETL allows standard operations of a transformation language but also other advanced features like

manipulation, navigation and query of both source and target model. ETL is a hybrid language that implements a mechanism composed by declarative definition of rules but also inherits the imperative features to handle complex transformation which can't be addressed with a declarative language.

ETL is similar to ATL because it is organized in modules (named EtlModule). A module contains a number of transformation rules. Each rule has a unique name and specifies both source and target model. A transformation rule can extend other transformation rules with different mechanisms (called lazy, primary or abstract). EtlTransformation defines a block statement in which is collocated the logic for populating the property values of the target model elements. ETL allows defining pre or post statements that can be executed before or after the transformation rule. [Kolovos, 2008]

Text-based transformations

The expression "text-based transformation" denotes a transformation which transforms an input document written according a text-format in an output document written according a different format. An example of text-based transformation languages are the XML transformation languages which allow transforming an XML document in another XML document or in a HTML document. Examples of XML transformation languages are XSLT, which is a W3C recommendation or Xquery which is a de facto standard used by Oracle, Microsoft etc.

Acceleo

Notwithstanding Acceleo is not properly a technology for model-to-model transformation, it has become a leading technology for generation of text from a model conformant both to Ecore based language and to annotated UML. It is based on the template-based paradigm in which the user creates text templates in which some parts can be dynamically defined on the basis of some model query. Since its launch, Acceleo has become a widespread solution in model driven engineering processes.

3.2.3 Model Based Testing tools

The previous sections have described several solutions to guide the choices of the meta-language on which a test generation process can be based. As stated in D501.010, this is a crucial choice because it affects the quality of the result of the developed brick. The testing process is based on "model-based" techniques.

Despite the generation of tests cases from a behavioural model of the SUT is mainly applied to functional black-box testing we want to apply it to gray-box testing, in order to cope with specific issues posed by critical systems. To model the system behaviour a FSM based DSML should be defined and proper tools supporting the modelling and testing phases should be developed. Here a brief survey of the most meaningful suites and environments is provided. Nevertheless many state-based testing tools are available. These technologies are supported by a large set of model based testing tools coming from both commercial, open source and academic contexts.

The broad spectrum of model-based testing tools makes difficult to cluster and compare them.

This section is focused on tools selected with two main criteria:

1. MBT tools that support a state-based language like UML or Finite State Machines (FSM);
2. MBT tools which provide automatic generation of abstract test case;

These two criteria are used for the construction of the table. In that table we consider other relevant features that enable us to compare the tools between them. These features are:

- If the tool is based on a DSML or on meta-model and in case which approach it follows;
- If the tool is commercial or open source;
- If the tool allows to simulate the model;
- If the tool allows to execute the test suite generated;
- The usability of the tool.

Table 3-1 summarizes and compares some model based testing tool selected with the criteria depicted before.

Tool	Based on DSML or meta-model	License	Simulation	Supported by graphical interface
Acceleo	EMF-Based	Open Source	Yes	Yes
AGEDIS suite	MOF-Based	Academic	Yes	Yes
Conformiq	EMF-Based	Commercial	Yes	Yes
GOTCHA	Not Specified	Commercial	Yes	Not specified
GraphWalker	Not Specified	Open Source	Yes	Yes
MaTeLo	Not Specified	Open Source	Yes	Yes
Nmodel	Not Specified	Open source	Yes	Yes
ParTeG	EMF-Based	Open Source	Yes	Yes
SpecExplorer	Other approaches	Commercial	Yes	Yes
Stateflow	Not Specified	Commercial	Yes	Yes
TestCast	Not Specified	Commercial	Yes	Yes
TestOptimal	Other approaches	Commercial	Yes	Yes
Torx	Other approaches	Academic	Yes	Not specified
Kermeta	EMF-Based	Open Source	Yes	Yes
Yakindu Statecharts	EMF-Based	Open Source	Yes	Yes

Table 3-1: State based modeling tool – a quick comparison

Acceleo

Acceleo is an open-source tool for code and test generation. Despite it is an implementation of MOFM2T standard, code generation is provided for EMF based models. This code generation language uses a template based approach in which there are dedicated parts where the text will be computed from elements provided by the inputs models. The dedicated parts are specified on the entity of the input models used to select and extract information from models. These expressions are often written with java implementation of OCL.

AGEDIS Suite

The AGEDIS tools is a suite developed within the AGEDIS project, an European project for the creation a methodology for automated model driven test generation and execution. The suite includes an integrated environment for modelling, a code generator for test generation, a runtime environment for test execution, and other feature like generation of report.

The input of AGEDIS tools is composed by three parts: the first is the behavioural model of the SUT, the second and the third are the testing directives which describe testing strategies and testing architecture of the SUT.

Both testing architecture and testing strategies are given by input using a UML modelling tools output. The UML modelling tools must be equipped by AGEDIS UML Profile. The behavioural model can be specified by UML diagrams like Class Diagram, Sequence diagrams, State Charts etc.

The AGEDIS suite has a model simulator which provides a feedback on the behaviour of the model and is used for model debugging. The test generator is based on two state-based test generators, GOTCHA and TGV, described below [Mishra, 2012].

Conformiq Automated Test Design

Conformiq Automated Test Design tool automates the design of functional tests for software and systems.

The tests generation produces tests from high-level system models without user intervention, complete with test plan documentation and executable test scripts in industry standard formats like Python, TTCN-3, C, C++, Java, and many others.

The generation starts from the model of the expected behavior of the real system from which Conformiq tools generate automatically human readable test cases, and executable test scripts.

Conformiq Designer is an Eclipse-based tool and is could be inserted within EMF category. Models can be described also with other third party modeler.

GOTCHA

The GOTCHA-TCBeans (which is the union of two previous tools, GOTCHA and Spider) is a framework developed by IBM designed to support development, execution and control of function tests using APIs and software written in Java, C, C++ [Farchi, 2002].

The main goal of GOTHCA-TCBean is to enhance testing activities with a systematic approach to test suite generation. The systematic approach allows higher functional coverage and exposes more defects early in the software development cycle. Main features of the GOTCHA-TCBean are:

- Creation and edition of models of SUT
- Simulation of models
- Generation of test suite from models
- Execution of the test suite
- Report and traceabili

GraphWalker

GraphWalker is MBT that allows both offline and online test sequences from Finite State Machines and Extended Finite State Machines.

The main features of GraphWalker are:

- it isn't based on UML but on FSM and EFSM (Extended FSM) and on a subset of UML rules, named GraphML, which is easier than normal UML
- it supports online test sequence generation. Using this feature it is possible to individuate a test path within model at runtime;
- it enables models reuse;
- the runtime environment allows models simulation;
- it hasn't start or stop points. This means that during the testing the same path is not chosen every test execution. The random variation will create a better 'test coverage' of the system under test.

Kermeta

Kermeta is a powerful meta-programming environment based on an object-oriented Domain Specific Language optimized for meta-model engineering. It provides specification of abstract syntax, formal semantics (OCL) and concrete syntax. It has model and meta-model prototyping and simulation.

Kermeta is fully integrated with Eclipse and includes features such as a compiler, an editor and various import/export transformations. It is available under an open source license EPL.

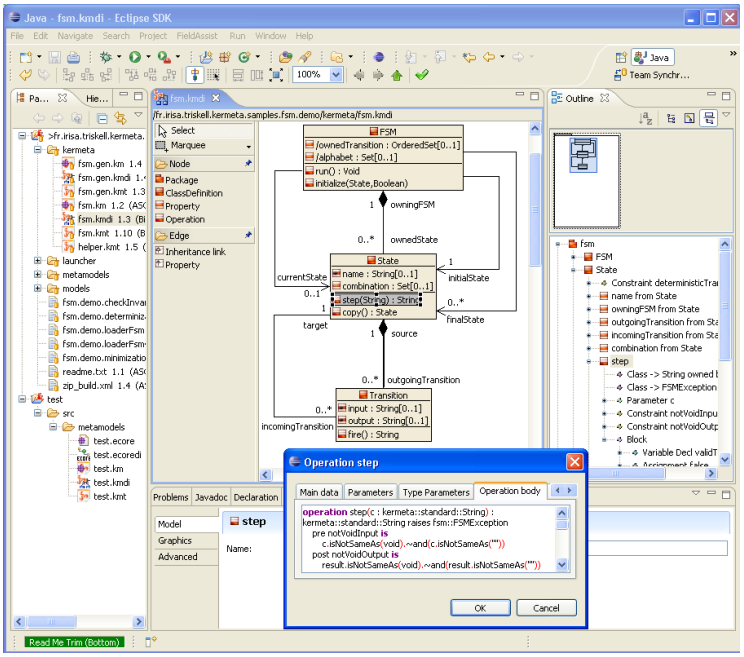

Figure 3-9: Kermeta

MaTeLo

MaTeLo (Markov Test Logic) is a test tool which uses the Markov Chains theory designed to reduce the costs and times of test phase of system or software. The usage model is made of transitions which represent the stimuli of the system, and the control to be applied to the output, representing the expected results. Transitions are separated one to the other through the use of states. States are stable states indicating the state of the system before the incoming of a stimulus.

Nmodel

Nmodel is a model-based testing and analysis framework based on C# languages. The Nmodel includes a library of attributes and data types for writing model programs in C#, a visualization and analysis tool named “Model Program Viewer”, a test generation tool and a test executor for both online and offline testing.

Despite the model is written with C#, to express scenarios, it is possible to write simple finite state machines (FSMs), then use composition to achieve scenario control during testing or to check temporal properties during analysis [Jacky, 2008].

ParTeG

ParTeG is a test generation tool based on algorithm that satisfy control flow-based coverage criteria which means that selected coverage criterion is transformed into a test model-specific set of test goals.

ParTeG is an Eclipse plug-in. It can be called from different views or diagrams: the state machine in the EMF tree view, from inside the state machine diagram, and from the file of the state machine diagram. Figure 3-10 shows a screenshot of it. A specific feature of ParTeG is the support to mutation analysis.

By means of ParTeG, one can create a list of mutants and can generate the corresponding test suite. It also uses an external tool, named Jumble, which measures the fault detection ability of test suite [Weibleder, 2013].

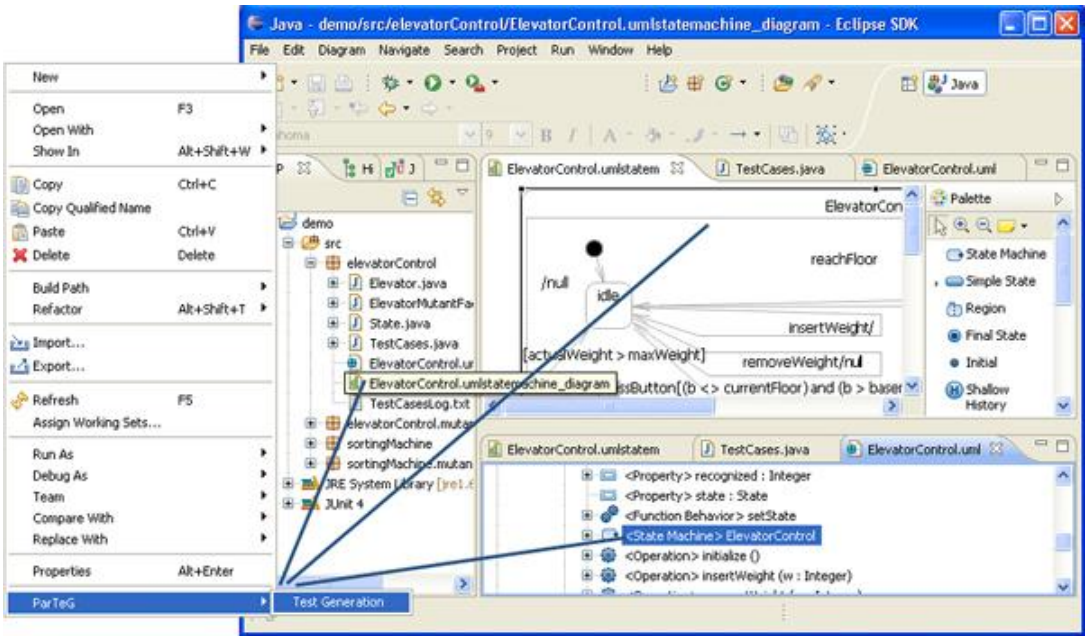

Figure 3-10: ParTeG Screenshot

Stateflow

Stateflow is a tool of Mathworks that, starting from the definition of FSM based model, allows the verification of the created model and code generation in several programming language. Stateflow is a widespread tool used for embedded software (in particular in automotive). It is based on a language with strong, formal semantics.

TestCast MBT

TestCast MBT is a tool for Model based testing which combines both automated test generation and execution. It allows defining coverage, generating tests, executing them and analyzing the results obtained.

The System Under Test is formalized into a UML Statechart model. That means that it is compatible with every tools that allow the modelling with statecharts. One of the most relevant features of TestCast MBT is the automated generation engine which automatically designs test cases from the formalized model of the requirements. As stated before, TestCast MBT provides an intuitive interface to compare actual test results to expected behaviours.

TestOptimal

TestOptimal is a web-based client-server tool that tests desktop and multi-tier enterprise applications. TestOptimal relies on a model described by using FSM creating interactively during the analysis of the web site under tested. Model can also be imported in GraphML formats. TestOptimal provides an XML based scripting language called mScript to connect the model to the SUT. TestOptimal automatically generates test adapter class skeletons where a tester can add function logic to run generated test cases.

SpecExplorer

Spec Explorer is a Model-Based Testing tool from Microsoft. Spec Explore is based on Spec#, a programming language with specification language features that allows defining a model describing the expected behaviour of a software system. From these models, the tool can generate tests automatically for execution within Visual Studio framework but also within many other unit testing frameworks.

SpecExplorer covers most of the relevant step of a MBT methodology. It allows creating a model of system and of the tests by using a State Machine which can be created, manipulated, explored by the tool. Furthermore it is possible to define many scenarios (e.g. degraded situations) in which generate and execute test cases. More reference can be found in [Utting, 2004] and [Campbell, 2008].

Torx

TorX is an automated tool for conformance testing. TorX generates the tests on-the-fly using a random strategy, which can be constrained by test purposes.

In TorX, automatic test generation and test execution are not done in separate phases but they are integrated. This is the key point of Torx because there is no complete test suite generated that is subsequently executed [Tretmans, 2003].

YakinduStatechart Tool

Yakindustatechart tool is a EMF-based environment for the specification and development of reactive, event-driven system based on the concept of statecharts. It combines four relevant features: it has a graphical editor for statecharts, it allows the validation of the created model on line during the construction, and it allows the simulation of the specifications described by model and allows generating code from model.

Despite this tool has not been created within the MBT aegis, it provides most of the MBT tools features. Below it is possible to see some screenshots of it.

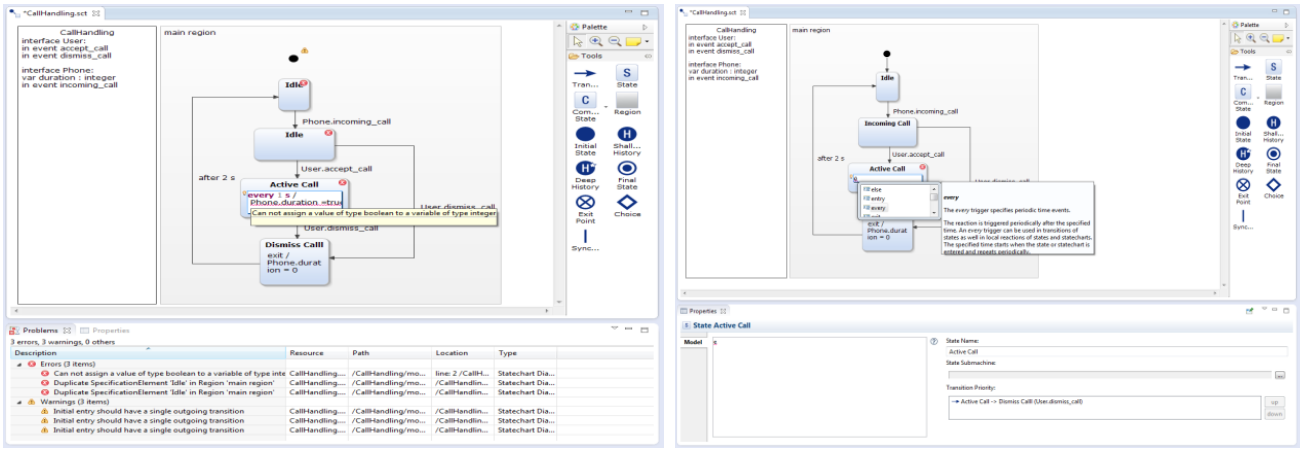

Figure 3-11: Yakindu Statecharts screenshots

Regarding the tool suite, the address is to adopt an open platform as Eclipse.

3.3 From Test Cases to Test Scripts

In a context where the importance of automation for the functional safety of all devices is increasingly questioned as the rail domain is, taking advantage from the automatic test generation would imply an important and concrete step towards an increase of railway system safety and reliability, due to its effectiveness in reducing human errors.

This is also underlined in a standards produced by CENELEC (the European Committee for Electrotechnical Standardization, which is responsible for standardization in the electrotechnical engineering field), the EN 50126 and EN 50128, which describe the processes and methods that are used to specify the most essential and important aspects for operability and safety in the rail domain (with a particular focus on software development).

In addition to the automatic test generation, there is also another critical point for the European railway industry: supporting the initiative backed by the European Union of creating a single Europe-wide standard for train control and command systems (ERTMS, see Par. 2.3).

A crucial step to guarantee and facilitate the interoperability of a trans-European high-speed rail system is the definition of an interoperable testing environment with standardized interfaces among the railway companies (in order to test several subsystems provided by different companies/suppliers), as stated in the “Communication from the Commission to the European Parliament and the Council on the deployment of the European rail signalling system ERTMS/ETCS” (04/07/2005), where UNISIG (the Union of Signalling Industry, which includes Ansaldo, Alstom, Bombardier, Invensys, Siemens and Thales) was asked a proposal for IOP (Interoperability) Testing.

In order to setup a multi-company interoperable testing environment, the adoption of a standard language is required. This language would allow different companies/suppliers to share all the steps through the definition of general test scenarios in a common language. The test step written in this general common language shall be properly understood by different system implementing proper adaptors. The usage of this standard language reduces the risk of misunderstanding/incoherence and enables the execution of interoperability tests in laboratory.

Obviously, due to the fact that each testing environment is built in its own language, all the companies/suppliers have to develop several adaptors for the interoperable testing environment (as depicted in Figure 3-12).

Hence, the ASTS tool chain in Crystal has to give the opportunity to automatically generate, starting from test cases, the test scripts, written in specific languages (either IOP or proprietary languages), by using specific plug-ins. For interoperability sake, the IOP *writer* has the highest priority in the Crystal project.

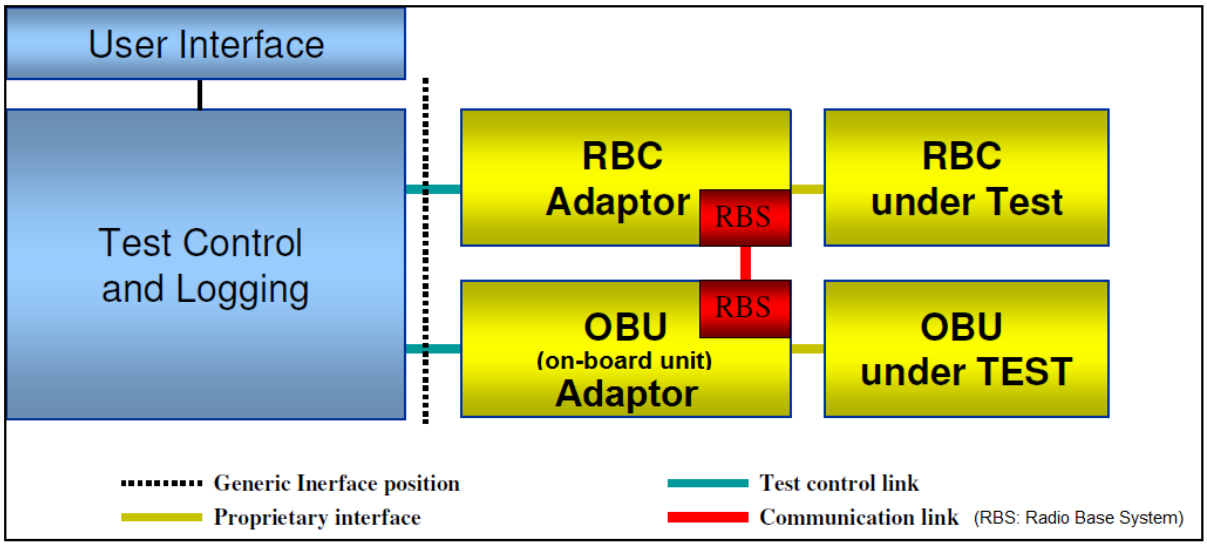

Figure 3-12: architecture overview of the test environment

3.4 Test Scripts Execution

This step is **external to the Crystal workflow** since it is based on proprietary technologies and simulators.

3.5 Test Reports and Feedbacks

This step could be performed by implementing proper tools which are able to interpret Test Log files and to visualize them in a user-friendly format. At the moment, the identification, starting from the not-passed tests, of requirements or parts of the system not rightly implemented requires a considerable effort from the system experts. On the contrary, thanks to the automatic traceability among requirements, test cases and test logs, it would be possible to pull out this information in an automatic way relieving the experts of that effort. Moreover, through the automatic generation of the report of the testing campaign, it will be possible to reduce significantly the effort currently spent in manually analyzing that report, overcoming all the current limitations affecting the entire process in terms of time and costs. Hence the supporting technology is that of the programming language.

4 Technology Functional Requirements

4.1 Requirements

After a thorough analysis of the use case chosen by ASTS, in particular focused on the use case goals, some technological requirements have been individuated. There requirements are listed in this paragraph and give a technological answer to the methodological requirements reported in the deliverable “CRYSTAL_D_D501.010 – Data and Methodologies report”.

In the following the list of requirements to which the technologies that will be adopted shall comply is summarized:

- **(REQ.501.020_01)** the adopted/produced artefacts shall use interoperable data format (possibly based on existing standards) in order to allow the implementation of automatic transformations;
- **(REQ.501.020_02)** the automated steps of the proposed methodology shall be properly implemented;
- **(REQ.501.020_03)** the modelling environment shall support the tracing of system requirements on model portions;
- **(REQ.501.020_04)** the test specifications shall trace the related system requirements;
- **(REQ.501.020_05)** the produced test cases shall trace the related test specifications;
- **(REQ.501.020_06)** the log analysis shall identify the satisfied/not satisfied requirements;
- **(REQ.501.020_07)** the involved artefacts shall be able to invalidate test cases after update of system requirements;
- **(REQ.501.020_08)** the adopted/produced artefacts shall support the multi-user utilization (cooperating with the RTP);
- **(REQ.501.020_09)** the adopted/produced artefacts shall support the versioning (cooperating with the RTP);
- **(REQ.501.020_10)** all the adopted/produced artefacts shall be verifiable, in compliance with CENELEC standard, since they shall be adopted in the lifecycle of railway systems;
- **(REQ.501.020_11)** the modelling environment shall allow future extensions of the implemented modelling language;
- **(REQ.501.020_12)** the modelling environment shall support the implementation of a new language which relies on a state-based formalism;
- **(REQ.501.020_13)** the modelling environment shall appear with a graphical interface and a palette which contains the appropriate constructs;
- **(REQ.501.020_14)** the test generation approach shall implement model checking techniques;
- **(REQ.501.020_15)** the test generation approach shall be able to generate test cases from model portions.

4.2 Industrial benefits of adoptable technologies

As deeply explained above, the automatic generation of the test cases is enabled by the application of the Model-Driven methodology. This methodology can be applied in the industrial context if a well-structured and semantically precise modelling language is defined in order to have a not ambiguous model of the system

under V&V. Between the different techniques able to define a new modelling language previously described, two of them are very appealing in industrial settings: those which rely on the UML and EMF technologies. The two methodologies are based on graphical approaches to model construction: this is necessary to better understand of the system under V&V, of the tests and to manage test suite (revisions, reused test, etc.). This paragraph explains the main industrial advantages given by the adoption of the EMF technology in the development of the new modelling environment.

The UML profiling technique allows extending UML from the semantic point of view by introducing newest concepts which extend those already present in the UML itself. The only constraint is that the new stereotypes introduced at the modelling language level must be mapped on UML meta-classes: for example, it is possible to extend the semantics of the UML State, defined in the State Machine context, to introduce the concept of states associated with watchdog timer, while it is not possible to introduce new syntactic concepts inside them. The main advantage in the usage of UML profiles is that they extend a language universally known and standardized in the international environment by OMG, hence the syntax of any model will be recognized in a standard way. In addition the syntax of the new modelling language should not be defined, but it is necessary only to identify the extensions you wish to add to UML. The disadvantage resides in the fact that UML is present as the basis of whatever UML profile: a modeler which wants to describe a system using the UML profile is not “bonded” to apply stereotypes on each UML element but she/he can model a portion of the system using the UML core; in this way it is very hard to set up a process based on automatic translations. Finally, if on the one hand, there are many modelling tools capable of working with UML and UML profiles, on the other hand is very hard to develop plugin able to extend them in an ad-hoc way.

The main advantage obtainable adopting the EMF technology, on the other side, resides in the simplest creation of graphical user interfaces of the modelling environment and in the highest customizability of graphical appearance of the model. The creation of new plugins for static analysis of the models and their simulation is also quite simple. EMF is widely adopted in industrial contexts of different application domains; the automatic generation of artefacts from models developed using the EMF technology is supported by various technologies currently integrated into Eclipse. The main disadvantage in the adoption of EMF is the non-standardization of the modelling language: a model developed using the modelling language that will be defined can be universally adopted within a company but it will not be written in a standard modelling language.

For these reasons and after a careful analysis of the state of the art in the lifecycle adopted in Ansaldo STS and in the railway domain, the EMF technology seems to be the most effective way able to implement the Crystal methodology: the entire usability on the tool chain will be higher than that obtainable by adopting UML profiles as well as the introduction of the Crystal process in industrial settings will be greatly simplified (in terms of start-up efforts, training activities, etc.). Moreover, considering that the V&V phases correspond to the ascending branch of a common life cycle “V”, there are no reasons which prevent the usage of UML and UML profiles in the early stages of descent branch (i.e. during the design and the development of a system); a step of automatic or semi-automatic translation can be inserted after the development in order to transform UML design models into V&V models written with the EMF technology.

For what concerning generation of test scripts and log analysis, it would be necessary to develop ad-hoc tools. In this sense the suitable technology would be the usage of general purpose languages (e.g. Java) which allow the implementation of required domain specific features.

4.3 Mapping requirements with the addressed technologies

Table 4-1 reports how the requirements given by the Use Case are mapped onto technologies listed in the previous Section. Each requirement shall be satisfied by one or more technologies, an empty cell (i, j) indicate that j-th technology does not impact on the compliance with the i-th requirement.

Requirement	EMF	Specific-purpose transformation languages (ATL / ETL)	Eclipse-based framework	General purpose language (e.g. Java)
REQ.501.020_01	EMF strongly relies on XML and XMI standard			Tools developed using general purpose language can implement standard data format
REQ.501.020_02		ATL allows to implement automatic generation of artefacts	Plug-ins allow the possibility to embed complex model transformation chains	Tools developed using general purpose language can implement the automated steps
REQ.501.020_03	EMF allows the annotation of traceability information			
REQ.501.020_04	EMF allows the annotation of traceability information			
REQ.501.020_05	EMF allows the annotation of traceability information	ATL allows to propagate traceability information		
REQ.501.020_06				Tools developed using general purpose language can implement strategies to individuate satisfied / not satisfied requirements
REQ.501.020_07	EMF allows the annotation of traceability information			Tools developed using general purpose language can read traceability information

REQ.501.020_08			Eclipse is featured with some plug-ins that allow interfacing with OSLC-compliant repositories	
REQ.501.020_09			Eclipse is featured with some plug-ins that allow interfacing with OSLC-compliant repositories	
REQ.501.020_10		The definition of high-engineered rules opens to the application of verification methodologies on model transformations. Moreover, the use of proven-in-use transformation engines is high recommended		Tools developed using general purpose language can be properly verified by current techniques
REQ.501.020_11	EMF has the capability to generate languages that can be easily extended	ATL superimposition is an example of technique that can be addressed when extending a model transformation		
REQ.501.020_12	Since its high versatility, EMF can be used to define state-based language			
REQ.501.020_13	EMF tooling is known to produce more easy-to-use solutions than other meta-modelling frameworks		Eclipse allow the adoption of the GMF technology to construct graphical interfaces	

REQ.501.020_14		ATL allows to implement proper transformation chains able to transform models into a concrete syntax of a model checker		
REQ.501.020_15		ATL can implement transformations which start from a portion of the source model		

Table 4-1: requirements-technology mapping

4.4 Taking charge of Methodological Requirements

This paragraph reports how methodological requirements have been taken in charge by technological requirements.

The list of methodological requirements is reported in the following: some of them (i.e. REQ.501.010_02, REQ.501.010_07, and REQ.501.010_09) are qualitative, their effective implementation in the Crystal project will be verified by ASTS after the implementation of the Crystal workflow:

- **(REQ.501.010_01)** the methodology shall be compliant with the lifecycle introduced by applicable norms, in particular it shall be applied at the system testing level (i.e. generated tests shall be used to perform the final validation against system requirements);
- **(REQ.501.010_02)** the methodology shall have an high level of automation, where possible;
- **(REQ.501.010_03)** the methodology shall trace the coverage between test cases, test reports and requirements;
- **(REQ.501_010_04)** the methodology shall support the consistency when different users work on the same system;
- **(REQ.501_010_05)** the methodology shall support versioning of the source models, of the test cases and of the test reports;
- **(REQ.501.010_06)** all the steps of the proposed methodology shall be verifiable since it shall be adopted in the lifecycle of critical systems;
- **(REQ.501.010_07)** the methodology shall be, whenever applicable, extensible in order to allow future extensions of the modelling language and of the final scope**;**
- **(REQ.501.010_08)** the modelling approach shall rely on state-based formalism as source modelling language;
- **(REQ.501.010_09)** the test case generation method shall be usable by experts of the rail domain which may not know how tests are generated;
- **(REQ.501.010_10)** the test case generation method shall not generate a test case from a test specification if the test specification is effectively infeasible, otherwise a test shall be eventually generated;

- **(REQ.501.010_11)** the test case generation method shall not re-generate test cases when no updates are performed on model portions.

For a detailed description of the methodological requirements please refer to the deliverable "CRYSTAL_D_D501.010 – Data and Methodologies report".

Table 4-2 shows the mapping between methodological and technological requirements. In particular the technological requirements are reported as rows while the methodological ones on the columns. The green cell (i, j) indicates that the implementation of the i-th technological requirement gives an answer to the j-th methodological requirement. The results showed by the table are that methodological requirements are completely covered by technological requirements and vice-versa.

	501.010_01	501.010_02	501.010_03	501.010_04	501.010_05	501.010_06	501.010_07	501.010_08	501.010_09	501.010_10	501.010_11
501.020_01		■					■				
501.020_02		■							■		
501.020_03	■		■								
501.020_04	■		■								
501.020_05	■		■								
501.020_06	■		■								
501.020_07			■								
501.020_08				■							
501.020_09					■						
501.020_10						■				■	
501.020_11							■				
501.020_12								■			
501.020_13									■		
501.020_14										■	
501.020_15											■

Table 4-2: methodological - technologic requirements mapping

5 Terms, Abbreviations and Definitions

API	Application Programming Interface
ASTS	Ansaldo STS
ATL	ATLAS Transformation Language
BNF	Backus-Naur form
CENELEC	European Committee for Electrotechnical Standardization
CO	Confidential, only for members of the consortium (including the JU).
CRYSTAL	**Cr**itical S**YST**em Engineering **A**cce**L**eration
DSML	Domain Specific Modeling Language
EBNF	Extended Backus-Naur Form
EFSM	Extended FSM
EMF	Eclipse Modelling Framework
EPL	Eclipse Public License
ERTMS	European Rail Traffic Management System
ETCS	European Train Control System
ETL	Eclipse Transformation Languages
FSM	Finite State Machines
GMF	Graphical Modeling Framework
GSM-R	Global System for Mobile Communications – Railway
HTML	HyperText Markup Language
IOP	Interoperability
M2M	Model-to-Text
M2T	Model-to-Model
MARTE	Modeling and Analysis of Real-Time and Embedded systems
MBT	Model-Based Testing
MDA	Model-Driven Architecture
MDE	Model-Driven Engineering
MOF	Meta Object Facility
MOFM2T	MOF Model to Text Transformation Language
OCL	Object Constraint Language
OMG	Object Management Group
OSLC	Open Services for Lifecycle Collaboration
QVT	Query/View/Transformation
R	Report
RBC	Radio Block Centre

SUT	System Under Test
UML	Unified Modeling Language
UNISIG	Union of Signalling Industry
UTP	UML Testing Profile
W3C	World Wide Web Consortium
V&V	Verification and Validation
XMI	XML Metadata Interchange
XML	eXtensible Markup Language
XSLT	eXtensible Stylesheet Language Transformations

Table 5-1: Terms, Abbreviations and Definitions

6 References

[CENELEC 50126, 2012]	CENELEC, *EN50126 - Railway applications - The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS) - Part 1: Generic RAMS* process; 2012
[CENELEC 50128, 2011]	CENELEC, *EN50128 – Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems;* 2011
[Harel, 2004]	D. Harel, B. Rumpe; *Meaningful Modeling: What's the Semantics of "Semantics"?;* IEEE Computer Society; 2004
[Clark, 2001]	T. Clark, A. Evans, S. Kent and P. Sammut; *The MMF approach to engineering object-oriented design languages*; In Workshop on Language Descriptions, Tools and Applications; 2001.
[Fuentes, 2004]	L. Fuentes-Fernandez, A. Vallecillo-Moreno; *An Introduction to UML Profiles*; The European Journal for the Informatics Professional, Vol. V, No 2; April 2004
[Steinberg, 2009]	D. Steinberg, F. Budinsky, M. Paternostro, E. Merks; *EMF: Eclipse Modeling Framework*; Addison-Wesley Professional; 2009
[Dai, 2004]	Z. Dai; *Model-driven testing with UML 2.0*; In Proceedings of the 2nd European Workshop on Model Driven Architecture, 2004.
[Jouault, 2006]	F. Jouault and I. Kurtev; *Transforming models with ATL*; In Satellite Events at the MoDELS 2005 Conference, pages 128–138. Springer; 2006.
[Obeo, 2013]	Obeo; *Acceleo*; http://www.eclipse.org/acceleo
[Kolovos, 2008]	D.S. Kolovos, R.F. Paige, F.A.C. Polack; *The Epsilon Transformation Language*; Theory and Practice of Model Transformations, Lecture Notes in Computer Science Volume 5063, pp 46-60; 2008
[MOFM2T, 2008]	http://www.omg.org/spec/MOFM2T/1.0/
[Mishra, 2012]	Mishra J.,Ali I.,Upadhyay A. K., *Automated Model Based Testing*, International Journal of Engineering Research & Technology (IJERT) Vol. 1 Issue 4; June 2012
[Farchi, 2002]	Farchi E., Hartman A., and Pinter S. S., *Using a model-based test generator to test for standards conformance*. IBM Systems Journal 41 pp. 89-110; 2002
[Campbell,2008]	Campbell, C., Grieskamp, W., Nachmanson, L., Schulte, W., Tillmann, N. and Veanes, M.; *Model-based testing of object-oriented reactive systems with Spec Explore*. Formal Methods and Testing 2008, LNCS 4949, Springer, pp. 39-76; 2008
[Basanieri, 2002]	F. Basanieri, A. Bertolino, E. Marchetti, *The Cow_Suite Approach to Planning and Deriving Test Suites in UML Projects*, Proc. Fifth International Conference on the Unified Modeling Language - the Language and its applications UML 2002, LNCS 2460, Dresden, Germany, September 30 - October 4, pp. 383-397; 2002.
[Jacky, 2008]	Jacky, J., Veanes, M.,Campbell, C., Schulte, W.; *Model-based Software Testing and Analysis with C#* ; Cambridge University Press; 2008.
[Weibleder, 2013]	Stephan Weibleder. *ParTeG (Partition Test Generator)*. http://parteg.sourceforge.net.
[Tretmans, 2003]	Tretmans J., Brinksma E. *TorX : Automated Model Based Testing.* First European Conference on Model-Driven Software Engineering, pp. 31-43; 2003.