 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

Specification, Development and Assessment for System Analysis and Exploration - V1

D603.011

DOCUMENT INFORMATION

Project	CRYSTAL
Grant Agreement No.	ARTEMIS-2012-1-332830
Deliverable Title	Specification, Development and Assessment for System Analysis and Exploration - V1
Deliverable No.	D603.011
Dissemination Level	CO
Nature	R
Document Version	V2.00
Date	2014-01-30
Contact	Andrea Leitner
Organization	VIF
Phone	
E-Mail	Andrea.leitner@v2c2.at

AUTHORS TABLE

Name	Company	E-Mail
Andrea Leitner	VIF	Andrea.leitner@v2c2.at
Jos Langen	Verum	jos.langen@verum.com
Valeria Vittorini	FEDII	valeria.vittorini@unina.it
Stefano Marrone	SUN	stefano.marrone@unina2.it
Arjan Mooij	TNO	arjan.mooij@tno.nl
Rubén Juan	ITI	rjuan@iti.es
Nadja Marko	ViF	nadja.marko@v2c2.at
Marco Bozzano	FBK	bozzano@fbk.eu
Alexander Hanzlik	AIT	alexander.hanzlik.fl@ait.ac.at
Matthias Tichy	Chalmers	matthias.tichy@cse.gu.se
Aleksander Lodwich	ITKE	Aleksander.Lodwich@itk-engineering.de
Gerald Stieglbauer	AVL	Gerald.stieglbauer@avl.com
Kurt-Lennart Lundbäck	ARCT	kurt.lundback@arcticus-systems.com
Dominique Segers	BARCO	Dominique.segers@barco.com
Christian Webel	Fraunhofer IESE	Christian.webel@iese.fraunhofer.de
Grischa Liebel	Chalmers	Grischa@chalmers.se

CHANGE HISTORY

Version	Date	Reason for Change	Pages Affected
1	20.12.2013	First harmonized version	

CONTENT

Content of Figures

Content of Tables

Introduction

1.1 Role of deliverable

This document aims to show the results of the first requirement elicitation phase for work package 603. Since this deliverable is due at a very early stage of the project, it mainly aims to introduce the different bricks in this WP and shows how they are planned to be used at this stage of the project.

This document can be understood as a living document. For some bricks, the use case needs have not been defined in detail until now. A detailed description will therefore be given in the next version of this document.

1.2 Relationship to other CRYSTAL Documents

This workpackage is quite complex and therefore has many relations to other deliverables. We will not list all deliverables here, but instead show how the bricks are related to use cases (UC).

Since bricks are always dedicated to one or more UC there is a natural relation between this document and the different UC description deliverables. This relation is described in Figure 0-1.

	SP2	SP3					SP4				SP5
	UC2.5	UC3.1	UC3.2	UC3.3	UC3.4	UC3.7	UC4.1	UC4.4	UC4.5	UC4.6	UC5.1
Task 6.3.1 Model-based system analysis						x					x
Task 6.3.2 System design and analysis with Sparx Enterprise Architect			x	x							
Task 6.3.3 Model-based requirements engineering		x			x						
Task 6.3.4 Architecture analysis and validation with the ASD:Suite		x									
Task 6.3.5 System design and analysis with AVL Cruise				x	x						
Task 6.3.6 Rapid design analysis (POOSL & NobiVR)							x				
Task 6.3.7 Design, analysis, and exploration using Mathworks Polyspace			x								
Task 6.3.8 System analysis using AbsInt			x								
Task 6.3.9 System design, analysis, and synthesis using Rubus ICE		x									
Task 6.3.10 System and performance analysis with DTFSim		x									
Task 6.3.11 Guaranteeing real-time execution of critical features								x	x		
Task 6.3.12 Interoperable architectural analysis										x	
Task 6.3.13 Functional and performance analysis									x	x	
Task 6.3.14 Scheduling requirement analysis	x										
Task 6.3.15 System analysis using ARTISAN Studio					x						

Figure 0-1: Relation between Tasks (Bricks) and UC using these bricks

1.3 Structure of this document

A brick in Crystal could either be a tool, a tool extension, or a methodology. The structure of this document takes this definition and is therefore separated in two chapters: Chapter 2 describes the activities on methodology bricks and Chapter 3 focuses on tool-related bricks.

The common structure for all brick descriptions is as follows:

1. General description
    a. For tools: the purpose and functionality
    b. For methods: state-of-the-art (SoA)
2. Application in CRYSTAL
    a. Requirements from the UC.
    b. What will be implemented/provided in the CRYSTAL project?
    c. How will this brick be integrated in the UC

2 Activities focusing on Methodology Bricks

2.1 B3.1 - Model-based system analysis and exploration

2.1.1 Description

Name:	Model-based system analysis and exploration (B3.1)
Contact:	Nadja.marko@v2c2.at
Dependencies	
License	
Additional information	

2.1.1.1 State of Practice analysis

Aim of the model-based system analysis task is the coordination and consolidation of activities within this work package in order to develop an overall Crystal system analysis and exploration approach including all engineering domains. In order to get the overall approach focused on industrial needs, we wanted to know the State of Practice (SoP) as well as the needs from industry regarding model-based engineering and model-based system analysis methods. An online survey was created for this reason which has been distributed to all Crystal partners, to partners from other EU projects, such as SafeCer[1], MBAT[2], VeteSS[3] etc, as well as to contacts from WP603 partners. The completed surveys are made anonymous and the results will be made available to all participants.

In this deliverable the survey and expected results are described. As the survey results have to be analyzed first, the detailed results of the survey will be part of the next deliverable.

2.1.1.1.1 Research questions to be answered

With the survey we want to answer some research questions. The main questions we wanted to have answered are:

- Which modeling techniques, modeling languages and modeling tools are used in industrial practice and why?
- In which phases of the software development process is model-based engineering used?
- How much time of the overall system and software development work is spent on model-based engineering?
- Which methods exploiting models are used for validation and verification?
- Which positive and negative effects result from the adoption of model-based engineering?
- What are the differences on the use and assessment of model-based engineering in different subclasses of companies and users?

[1] http://safecer.eu/
[2] https://www.mbat-artemis.eu/
[3] http://vetess.eu/

2.1.1.1.2 Survey content

The survey was developed by partners from ViF, CTH, ITK, Verum, FBK, FED-II, and SUN. Every partner proposed questions which have been of interest for him. As a result an extensive list of questions was obtained which has been harmonized and shortened with respect to the research questions to be answered. The final outcome is a survey that consists of 24 questions and should take approximately 15 minutes to answer. It targets on software architects, developers, project managers, system engineers, etc. from OEMs and suppliers from the embedded systems domain.
Questions are asked providing possible answers (multiple choice, single choice, ratings) as the survey should not take too much time. Nevertheless, the answers are not limited to the provided answers. The survey participant can fill in his own answers as well.
Mainly, the survey consists of 4 parts:

- General context
- Model-based engineering background
- Applied approaches (SoP)
- Advantages/challenges of model-based engineering

The complete survey can be found in Annex I.

2.1.1.1.2.1 General context

First part of the survey is target at the context of the survey participant. It should give us information about the domain, the products being developed, the company size as well as the working tasks of the participant. Context questions are for example:

- Is your company a Small and Medium-sized Enterprises (SME, <= 250 employees) or a large company (or part of)?
- In which domain do you work?
- What are your main working tasks?

2.1.1.1.2.2 Model-based engineering background

In addition to the general context questions, some context questions regarding model-based engineering activities are asked. With these questions we want to find out the experiences of both the survey participant and the company regarding model-based engineering. Questions of this part are for example:

- Please rate your experience with model-based engineering.
- What is the product you are targeting with model-based engineering?
- How relevant were the following reasons for introducing model-based engineering in your division/department?
- In which phases of the development process are you using model-based engineering?

2.1.1.1.2.3 Applied approaches

In order to get an impression of the SoP, the applied approaches (modeling languages, methods and tools) are asked with questions such as:

- Which modeling environment do you use personally and which one is used in your division/department?

- Which modeling language(s) do you use personally and which one(s) are used in your division/department?
- How would you compare your usage and the usage within your division/department of model-based and non model-based tools for performing engineering activities?
- For which purpose does your division/department currently use models and what do you personally think models should be used for?

2.1.1.1.2.4 Advantages and challenges

The last part of the questionnaire addresses the advantages and challenges that come along with model-based engineering. With this part we wanted to gain insight into shortcomings which could be subject for improvements within the Crystal project:

- What were the effects of introducing model-based engineering in your division/department?
- To what extent do the following potential shortcomings apply to the applied modeling approach?

2.1.1.1.3 Expected results

The survey results should give information about cross-domain needs regarding model-based engineering. This should include needs about:

- Modeling approaches
- Tools
- Data integration mechanisms

The results should help us to guide activities within WP6.3. More detailed results will be part of Deliverable D_603.012.

2.1.1.2 Model-based system analysis using model-driven techniques and gray-box testing

The dynamic verification (by simulation) that a system in its whole behaves as expected, i.e. it respects its functional requirements (including safety related ones), is known as functional testing or behavioral testing [Myers, 2004].

Model-Based Testing (MBT) is mainly used to generate functional tests. It is a testing approach based on the construction of an accurate model both of the system under test (SUT) and of its external environment, which is derived from the requirements specification. MBT is usually considered a form of black-box testing, because tests are generated from a model and information about the internal structure of the SUT is not used [Utting, 2007].

The main advantage of functional testing techniques is that they are relatively easy to implement; the main disadvantage consists in the difficulty of balancing test effectiveness and efficiency.

As effectiveness is difficult to predict, a thorough and extensive (thus costly and time consuming) test specification and execution process is usually performed on critical systems. Given the high number of variables involved, the required simulations (or test-runs) are prohibitive; thus the process is necessarily either unfeasible or incomplete, with possible risks on system safety. It can be proven that exhaustive black-box testing is impossible to achieve with no information about system implementation [Myers, 2011].

Test adequacy can only be assessed by means of empirical techniques, e.g. when errors/test curve flattens out [1020 WG, 1987]. Another important - though often neglected - limitation is that black-box testing approaches are based on a system specification, which is usually expressed in natural language and destined to be corrected, integrated and refined several times during system life cycle. Therefore, its completeness and coherence are far to be guaranteed, and this is especially true for complex systems.

This is because gray-box approaches are necessary for critical systems. Gray-box testing approaches support functional testing in allowing test engineers to fine tune the test-set with the aim of an effective coverage of functionalities with the minimum effort in time. The result is a significant reduction in test-set complexity while maintaining or improving test-effectiveness.

On the other hand, the model driven way could become a new paradigm for systems development in industrial settings, since it could have a strong positive impact in reducing time to market and improving the quality of the product. Model Driven Testing (MDT) addresses the optimization of the verification process by bringing Model Driven concepts and techniques into testing. Nevertheless, these two practices are not fully integrated, they are not really perceived as the two sides of the same coin.

MDT is sometimes considered synonymous of “Model Based Testing using UML”. Using UML diagrams (state machine diagrams, use-case diagrams, sequence diagrams, etc.) to generate test cases is just one of the well-known model-based testing techniques, but MDT could be more than this. MDT enables the definition of processes for the automatic generation of test cases, based on model transformation techniques, and links may be also defined between the MDT activities and the Model Driven Architecture steps in the development cycle of the system [Dai, 2004].

UNIFEDII and SUN will develop in Crystal a methodology brick addressing the V&V activities in the development of safety-critical embedded systems. Specifically, the methodology will support the automatic generation of test cases for gray-box testing at system level by using model-based and model-driven techniques. General issues about the modeling approach, generation techniques, and languages are addressed in the context of WP603 and described in the present document. Their instantiation to the UC501 is performed within WP501; the development of the methodology will be carried out within WP612. The brick will be integrated in the UC501.

Hence, the final contribution of UNIFEDII and SUN to this deliverable document is twofold:

- A methodology for test cases generation is defined and described, starting from an initial description of its definition process;
- A background related to the main themes addressed by the research activities is provided; a state of the art on specific topics is also planned, taking into account the consolidated survey results and the availability of input from WP501 and WP612.

2.1.1.2.1 Test Cases Generation Methodology: overview of the definition process

The main macro-activities that we have planned to perform in order to meet the goal are introduced in Figure 2-1. The concrete development of them depend on some choices made on the basis of the input provided by the partners mainly involved into the UC501 requirement specifications, as well as the implementation of the tool-chain (interoperability issues).

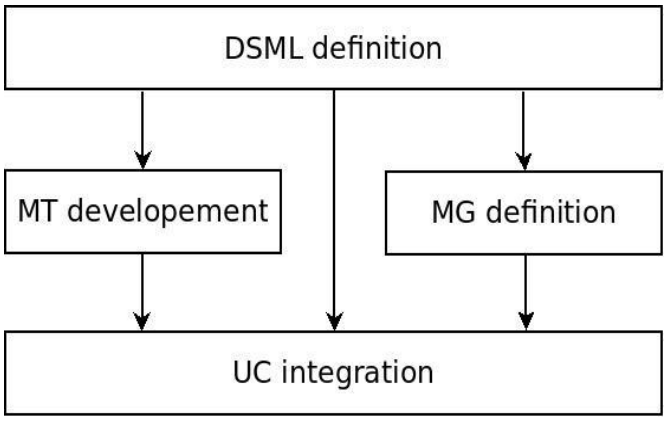


Figure 2-1: Schema of macro-activity phases

A first macro-activity (*DSML definition*) requires that a modeling language is defined to represent the system behavior and the test properties. The language must allow for easy modeling of a set of domain specific concepts from UC501 or from the application domain of interest (i.e. it has to be a Domain-Specific Modeling Language (DSML)); it must have a formal semantics in order to avoid ambiguities and to provide a basis for the application of formal verification techniques. It is a state-based language as the system behavior and requirements of a critical system are usually specified in the form of state-transition machines and/or tables. The meta-language to be used in order to define this language must meet the interoperability requirements.

The second macro-activity (*MT development*) requires the definition and implementation of proper Model Transformations (MT) in order to generate the artifacts needed in the test generation step from the models represented by means of the DSML. Thus, the target language(s) has (have) to be identified, the sets of transformation rules have to be defined, and the transformations must be implemented according to the selected test generation techniques. The technology used in developing the MT is also chosen on the basis of interoperability issues.

The third macro-activity (*MG Definition*) concerns the definition of *Modeling Guidelines* (both for system and test properties). The guidelines aim at supporting the system modeler in understanding the best way to represent system aspects; on the other hand the guidelines will support the test engineer in specifying the input for the automatic test case generation process. This macro activity will explore the state of the art in the definition of best practices, patterns & anti-patterns and model development methodologies.

Finally, the fourth macro-activity (*UC Integration*) addresses the integration of the resulting model driven process and tools into existing and assessed V&V processes as they are adopted by industries. This activity will first study the integration according to the three levels previous defined (DSML language, MTs and modeling guidelines). Some of the topics that will be investigated are: requirements traceability, coverage measurement, support to test execution, log analysis and reporting.

The oriented arcs in Figure 2-1 represent the input/output relationships between the macro-activities. Hence *DSML definition* has to be completed before starting *MT development* and *MG definition* which, in turn, may be performed in parallel and both have to be completed in order to perform *UC integration*.

With regards to these research activities, here a birds-eye view is provided of the mainstream background topics: Gray-Box Testing, Model Based Testing, Model Driven Testing and Techniques and Automatic Test Generation Techniques. Future versions of this document will provide a deeper state of the art about specific issues, centered on topics which will be relevant in the development of the macro-activities.

2.1.1.2.2 Test Case Generation Methodology: background.

This section describes the background of the research activities, namely Gray-Box Testing, Model Based Testing, Model Driven Testing and Techniques and Automatic Test Generation Techniques.

This Section is intended to be the reference for concepts, methods and approaches we need to cite or refer to in the development of the generation methodology. As we make large usage of concepts from model-driven engineering, a special attention is given to model-driven terminology, concepts and techniques.

2.1.1.2.2.1 Gray-Box Testing

Since early years, the software engineering community has addressed several approaches in testing complex software and systems. One possible classification is the one that focus on the degree of knowledge the system tester has on the internal dynamics of the SUT [Meyers, 2004]. Such classification mainly detects two strategies: the black box-testing (also called input-output testing) and the white-box testing (also called logic-driven testing) strategies. The first is based on a “zero-knowledge” approach considering a complex system as an opaque box and testing the SUT by verifying that the expected and the actual output of the systems given a generic input are equals. On the other hand, white box testing concentrates on the internal structure of the SUT focusing on the verification of the internal behavior with respect to the requirements: it can be defined as a “full-knowledge” approach.

Gray box testing approaches combine the advantages of white-box and black-box tests strategies. Reason for adopting gray-box approaches in V&V processes have been explained in the introduction.

Several gray box testing approaches are present in the scientific literature focusing at different testing levels (i.e. the kind of software artifacts under test: unit, integration, system or user/acceptance testing).

While the knowledge of white-box testing is clear (for software intensive systems it is the source code), it is not clear what the form/nature is of knowledge at the base of gray-box testing approaches. There are a lot of heterogeneous approaches in the scientific literature: some of them start from a formalized specification of the system or of its components [Baharom, 2008], others are able to generate test cases for software starting from contract-based specification techniques [Dadeau, 2011], and others exploit Finite State Machines [Petrenko, 1995].

In the rest of this paragraph, we focus on *system level testing* and in particular on how gray box testing improves the quality of system testing still remaining a feasible solution. System testing “is concerned with testing the behavior of an entire system” [Abran, 2004]. Effective unit and integration testing will have identified many of the software defects”. Notwithstanding, unit and integration testing are often conducted by means of invasive methods (drivers and stubs are some the most common of them); the rationale for system testing is the necessity to have a level of testing where all the components interact in the same way in which they would interact during the operational phase of the software/system.

For this reasons, system level testing has been traditionally accomplished by means of black-box testing. This approach expresses its limitations in case of safety-critical systems; in fact limiting to the observation of the mere interfaces can prevent the testing engineer to observe the possible passage through some hazardous states. International safety and quality standards prescribe methodologies and techniques to use in order to achieve to high integrity systems [IEC, 1998], [CENELEC, 2004]. For many of such standards, system level testing phase with a limited or full knowledge of the system is often mandatory since black-box techniques. Complex systems, on the other hand, can have internal emergent dynamics that must be known and controlled during the testing phase and system testing is the first moment when these dynamics start to appear.

Such problem is worsening in case of critical systems where emergent behaviors can bring the system into unexpected hazardous states: thus black-box testing is considered as not sufficient to eliminate all the possible system hazards. Some approaches are present in literature of gray-box testing of (safety-) critical systems [De Nicola, 2005], [Piper, 2012]. Since UNIFEDII and SUN will contribute in Crystal developing V&V methodological bricks mainly inside the UC 501, gray box testing approaches are of paramount importance

due to both the safety critical nature of the system and the system level of the testing activities to perform inside this UC.

In order to accomplish perform gray box (system) testing, hence having “limited knowledge” about the SUT, a model of the system behavior is needed: indeed, gray box testing approaches are quite always ***model based-testing approaches***, too. Many approaches that exploit models for gray box testing are present in the scientific literature, for example in [Linzhang, 2004] where UML activity diagrams are used to generate test cases, [Petrenko, 1995] where FMS are used to understand about the internal behavior of the system, [Dong, 2009] where architectural models are used (AADL language).

2.1.1.2.2.2 Model-Based Testing

The term “model-based” applied to testing is used with several meanings [Utting, 2007]:

- Generation of test input data from information about the domains of the input values (domain model). This reduces hand-made work but does not provide any information to know whether a test has passed or failed;
- Generation of test cases from an environment model. A model represents the environment of SUT. This approach does not provide information on testing outcome, too;
- Generation of tests cases from a behavioral model. A model describes the expected behavior of the SUT. This approach needs oracle information to compare outcomes;
- Generation of test scripts from abstract representation of tests. Models describe test cases. From them proper transformations generate machine-readable tests.

Our research activities focuses on the third meaning of model-based testing, but in Crystal we address the modeling, the test requirement and the test generation phases, according to the general schema in Figure 2-2. We do not deal with oracles and test execution. The generation of tests cases from a behavioral model of the SUT is mainly applied to functional black-box testing. As already explained, we want to apply it to gray-box testing, in order to cope with specific issues posed by critical systems.

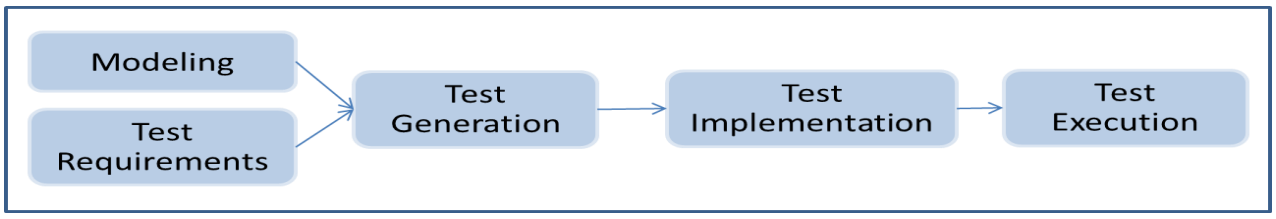


Figure 2-2: MBT phases

Several taxonomies and surveys have been published on MBT; they provide a deep inside of the most meaningful model-based methods and approach according to possibly different point of views. We cite some works and collections which are representative of four different perspectives which are pertinent to our research activities: general review, tools, embedded systems, and so called “model-driven testing”. Perhaps the most famous taxonomy of MBT is the one provided by Utting, Pretschner, and Legeard [Utting, 2012]. It is broad taxonomy based on three classes: 1) Models, i.e. the models applied in the MBT process; 2) Test Generation, i.e. the approaches on which the test generation process is based, depending on the test selection criteria, generation technology, and the expected generation results; and 3) Test Execution, i.e. the execution options, depending on the test platform.

The book authored by Utting and Legeard [Utting, 2007] focuses on model-based testing tools to generate test suites and the practice of functional black-box testing.

The collection in [Zander, 2011] contains a work from Zander, Schieferdecker, and Mosterman which proposes to extend this taxonomy in the context of embedded systems and several case studies from different domains as medicine, automotive, control engineering, telecommunication, entertainment, and aerospace. A survey on model-driven testing techniques is in [Mussa, 2009]. Indeed, model-driven testing is considered to be a kind of model-based testing which makes use of UML and model transformations. Some clarifications must be given to avoid that ambiguities may arise in adopting the model-driven terminology.

2.1.1.2.2.3 Model-driven techniques

Model-Driven Engineering (MDE) is a promising approach that is able to cope with the increasing complexity of platforms. More in general, model-driven techniques bring together two important aspects:

- DSMLs, which formalize the application structure, behavior, and requirements within a specific domain;
- Transformation engines and generators that analyze certain aspects of models and synthesize different types of artifacts, such as source code, simulation inputs, XML deployment descriptors, or representation of models.

The MDE initiative proposes a process definition wider and not limited to the development as for other approaches (Model Driven Architecture [OMG, 2003], Model Driven Software Development [Mellor, 2003]). MDE methods and techniques are applicable to general purpose software systems as well as to critical systems. In particular, in this last case, the effort must be oriented to support the qualitative and quantitative analyses since the verification of system properties plays a crucial role for the system success. Great benefits may derive from the adoption of MDE in the development of critical systems: the paradigm of systems design “construct-by-correction”, typical of processes based on testing and verification of the late-time properties of a system, can be replaced by paradigms “correct-by-construction” where, by verifying the correctness of both initial system model and model transformation, one can assure the correctness of the final model. The effort of verification can thus be concentrated in the initial stages of the system lifecycle.

Within the quantitative evaluation, outlined above, and within the scope of the techniques and methods of the MDE, a first scenario of application of these techniques to critical systems is clearly defined: there is the possibility to increase the spread of formal methods in industrial development processes in real systems. In fact, the ability to define high-level languages closer to the user (for abstraction and ease of use) as well as the ability to generate automatically models (formal models for quantitative analysis) from the first allows the usage of formal methods in a “transparent” way.

Domain-Specific Modeling (DSM) means a set of procedures and modeling artifacts that are specific to a given application domain [Kelly, 2008]. They are different from existing general purpose techniques since they directly use concepts in the modeling that belong to the application domain. These concepts and methodologies have been introduced to increase the level of abstraction with respect to the current programming languages. With the term application domain is intent both a technical domain such as persistence, user interface, communications, transactions, and a functional domain as a business domain of telecommunications, banking, insurance or retail sales. In particular the formers can be addressed as “horizontal” domains while the latter as “vertical”. In practice, each DSM solution focuses on very small domains because, in doing so, one has a better chance to automate procedures.

At the base of a DSML approach is the definition of a language. A language provides an abstraction for the development and it is the most visible artifact for developers. In DSM it is used to define specifications that manual programmers would treat as source code. If the language is built correctly, it allows one to apply terms and concepts of a particular domain. This means that a domain-specific language is probably useless in other domains. For domain-specific languages, the same definitions that are adapted to languages in general can be applied. The modeling languages are composed of syntax and semantics. On the syntax, one can further distinguish between abstract and concrete syntax. The first indicates the structure and

grammar rules of the language, while the second defines symbols of the notation and form of representation of the language. To increase the abstraction and to generate more complete code, it is usually necessary to extend both syntax and semantics.

The Unified Modeling Language (UML) [OMG, 2011] is a well known general purpose standardized modeling language for software system specifications. UML can be extended through the profiling mechanism that allows customizing UML for a particular domain or platform; this mechanism is defined in the UML Infrastructure. The UML profiling is actually a lightweight meta-modeling technique to extend UML, since the standard semantics of UML model elements can be refined in a strictly additive manner. Since this extension mechanism is part of the standard UML, it can be supported by the tool for UML. This feature is one of the main advantages of UML profiles compared to the other mechanisms of customization of UML that are not part of the standard UML and therefore are not supported by the UML tool. Another important advantage of the profiling mechanism UML is that it avoids redefining concepts already defined in UML. A UML profile is represented by a UML package stereotyped by tag “profile”. There are three extension mechanisms used to define a UML profile: stereotypes, tagged value and OCL rules [Fuentes, 2004]:

- stereotypes are the main constructs for the specification of a UML profile. A stereotype is a particular type of UML class (actually it is a specialization of the class meta-class from the UML meta-model);
- tagged values are properties (specialization of the property meta-class) that belong to one or more stereotypes;
- OCL rules are defined by the Object Constraint Language [OMG, 2012]: they express constraints the profiled models must be subject to.

UML can be extended through the profiling mechanism that allows customizing UML for a particular domain or platform; this mechanism is defined in the UML Infrastructure. The UML profiling is actually a lightweight meta-modeling technique to extend UML, since the standard semantics of UML model elements can be refined in a strictly additive manner. Since this extension mechanism is part of the standard UML, it can be supported by the tool for UML. This feature is one of the main advantages of UML profiles compared to the other mechanisms of customization of UML that are not part of the standard UML and therefore are not supported by the UML tool. Another important advantage of the profiling mechanism UML is that it avoids redefining concepts already defined in UML. A UML profile is represented by a UML package stereotyped by tag “profile” [Giachetti, 2009], [Selic, 2007].

Transformations of models are a key concept in the MDE context. They allow obtaining a model automatically from another, for example they are useful to change the formalism and to generate a formal model starting from a UML model. 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 modeling phase, and then reducing it for a specific platform, during the development stages. Model transformations can be grouped into two categories.

Model-to-Model (M2M) transformations: M2M transformations aim at transforming source models in other models, also expressed in different formalisms. The main motivation of their need is that the new model enables to perform analyses that are not feasible in the previous formalism. An example of language used to write M2M and Model-to-Text (M2T) transformation is the ATLAS Transformation Language (ATL) defined in the ATLAS Model Management Architecture (AMMA) platform [Jouault, 2006]. ATL is a hybrid language, i.e. it is both a declarative language and imperative one.

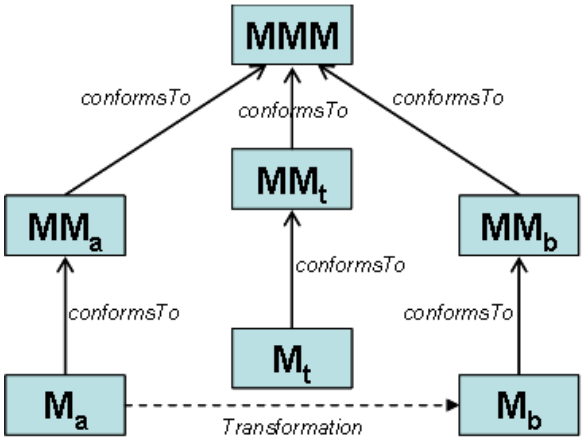


Figure 2-3: Model Transformation schema (from http://help.eclipse.org)

In the pattern depicted in Figure 2-3, a source model Ma is transformed into a target model Mb, according to the rules defined in the transformation Mt. The transformation can be seen as model since it is software. Source and target models, as well as the transformation definition are conforming to their respective meta-models: MMa, MMb and MMt. All meta-models in this example are conforming to MOF meta-meta-model (obviously this relationship is not strictly necessary; other meta-meta-models are of course usable). This schema is general enough to be adopted by all other transformation languages. ATL, as mentioned before, is a mixed language which contains declarative and imperative parts, nevertheless, the ATL philosophy encourages the use of the declarative style in specifying transformations. However, sometimes it is difficult to provide a solution completely declarative in a transformation problem. In this case it is possible to use characteristics of the imperative language. ATL transformations are unidirectional, source models are read-only and the transformations produce write-only destination models. The implementation of a bidirectional transformation makes it necessary to realize a pair of transformations, one for each direction.

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 as example into an Ecore based language, can be translated into a specific data format understandable by existing solvers. M2Ts can be divided into two categories according to the constituting principles:

- Visitor-Based Approaches: the source model is explored and, during the exploration, text is serialized into an output channel (a file). An example is constituted by ATL query [Jouault, 2006];
- Template-Based Approaches: the text is organized into templates where “hot-spots” (points that are subject to change according to model structure or values) are calculated by query on the model itself. A widespread example of this technology is constituted by Acceleo [Obeo, 2013].

2.1.1.2.2.4 Model-driven testing and model-driven test techniques

MDT is the application of Model Driven Architecture (MDA) principles to system testing. According to MDA the system is first specified from the functional point of view without any reference to the platform on which it will be deployed. This model is named Platform Independent Model (PIM). Using a transformation language PIM is transformed into a Platform Specific Model (PSM) which contains more detail of technology platform.

Finally, source code is generated from PSM using transformational rules. According to MDA, models are the main artifacts which guide the whole system development cycle.

MDT applies the same approach to testing activities. Two levels of transformation steps are present: vertical transformations and horizontal transformations. Vertical transformations are defined inside the test design process, from an abstract test description, named Platform Independent Test. This kind of transformation allows obtaining a Platform Specific Test (PST) and to generate a test code for a specific technology platform.

Horizontal transformations are defined across the system and the test design steps. This is a very innovative point because horizontal transformations allow performing testing activities early in software development cycle. The first transformation step builds an abstract description of the test suite from an abstract description of the system, without its platform specific model. The transformation between PSM and PST has the same rational; it allows producing a concrete test suite before the system code is available. Figure 2-4 summarizes the two-way approach of MDT taken from [DAI, 2004].

Hence, MDT has two main advantages: 1) testing activities may start early in the development process, because a platform independent test is produced with (or even before) PIM; 2) testing activities are better supported by tools and languages during the whole development cycle.

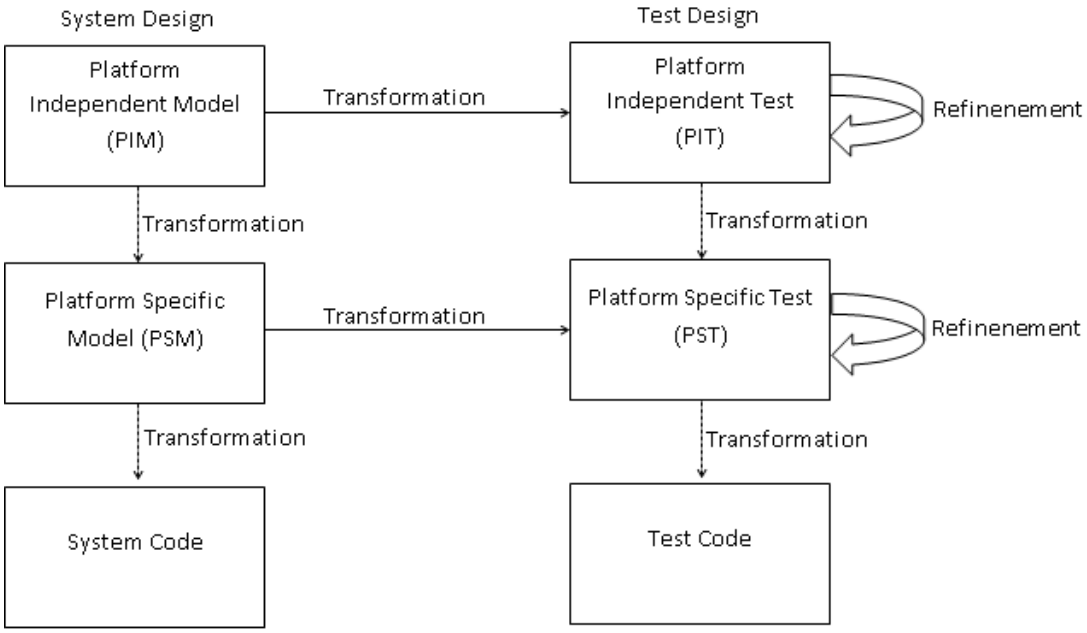


Figure 2-4 - System Design vs. Test Design [Dai, 2004]

A common language used to support MDT based approaches is UML and its extensions. An example is the *UML Testing Profile (UTP)* from OMG. UTP introduces concepts like test components or test control which are used to realize the test behavior of the system.

Some approaches are able to generate a high level representation of a test suite (and then executable test code) starting from an UTP model of a SUT. MDT has been used in several industrial domains. In [Hecker, 2003] an example of the application of MDT to web-based distributed services architecture is described and [Guelfi, 2008] shows an interesting application of MDT within the automotive domain. This work describes an application of MDT to the testing of the safety control of airbag systems. Despite these and other examples, the application of MDT is far to being considered an assessed practice in industrial settings.

Several model based approaches are founded on techniques taken from MDE. Even if they are often considered as “model driven testing”, they just use one or more MDT features, such as DSMLs, UML profiling, model transformations, etc.

We distinguish between the MDT process described above and such approaches we refer "*model driven testing techniques*" since they are a sub-set of MBT.

Some relevant works show the advantages of model-driven testing techniques within software development. One of the most common ways to establish MDE principles in software development is the use of UML diagrams (like Class Diagrams, Sequence diagrams etc); they are often associated to OCL constraints in order to generate automatically executable test cases. An example of this approach can be found in [Bouquet, 2007] in which the authors use a subset of UML diagrams constrained with OCL to generate test cases. The use of OCL constraints is necessary to prevent ambiguous behaviors. This point is very important in model driven approaches because it is necessary to have a clear understanding of syntax and semantics of source and target models in order to be able to perform model transformations [Sendall, 2003].

Another language often used is Testing and Test Control Notation version 3 (TTCN-3) which is a strongly typed test scripting language used to automatically determine whether a system fulfils its requirements. An example of TTCN-3 applications in testing can be found in [Tomasson, 2013] in which MDE principles are used for testing in ICT domains.

A wide literature can be found that describe model driven techniques in test automation. [Javed, 2007] proposes an approach using sequence diagrams. In this approach, the authors start with modeling the system using sequence diagrams and they use a chain of M2M and M2T transformations to obtain test cases. [Crichton, 2007] also starts from UML models to generate test cases. In [Mingsong, 2006] an approach is presented to test case generation from UML activity diagrams: in this approach the authors randomly generate abundant test cases from a SUT written in Java code. By running the program with the generated test cases the corresponding program execution traces are obtained, they are compared with the activity diagrams according to specific coverage criteria to build a reduced test case set which meets the coverage criteria. A survey of model driven testing techniques can be found in [Mussa, 2009].

2.1.1.2.2.5 Test Case Generation Techniques

In the following test case generation techniques existing in the literature of model based approaches are overviewed. A good taxonomy of model based testing approach for embedded systems is present in [Zander, 2011]. Figure 2-5 graphically depicts this taxonomy.

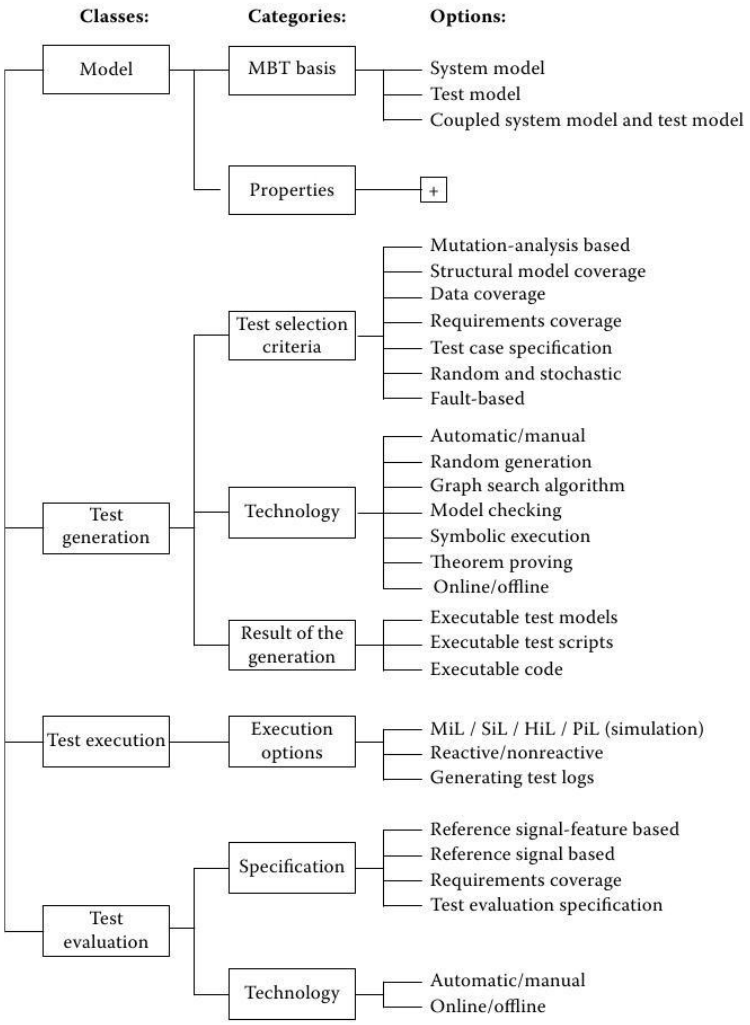


Figure 2-5: Model Based Testing overview [Zander, 2011]

In the rest of this contribution, the focus is set on test generation technology giving further details on some of these techniques:

- Random generation: this approach targets on the production of random test data as input for the SUT. The benefits of random testing techniques are mainly its inexpensiveness, its capability to be applied in the evaluation of software reliability level and its capability to be exploited to perform stress testing. The disadvantages are that there is no assurance for full code coverage and that it needs an automatic log verification phase due to the huge size of produced logs;
- Path-Oriented Methods: this approach generates test cases from control flow graphs or finite state machines by finding paths in models. The analysis could be static or dynamic. In first case test case generation is made without program execution. A well-known technique in this approach is symbolic execution. Symbolic execution executes a program using symbolic values of variables instead of actual values. With this method it is possible to obtain inequalities that describe the conditions necessary to cross the path. In general, path-oriented testing is NP-hard, but with linear constraints it's possible to use linear programming techniques. The way to overcome these problems is to use a dynamic approach in which the analysis is made during run-time of program under test. Program execution flow is monitored during run-time and, if there are some deviations from expected flow, some heuristic or meta-heuristic techniques like simulated annealing [McMinn, 2004] or backtracking are used to identify the problem. The major disadvantages of this technique derive from program execution because many iterations are often needed to generate a test suite;

- Model checking is a technique used to analyze a finite-state representation of a system for property violations. The main component, a model checker, analyses all reachable states and if it detects no violations, then the property will be true. Instead, if the model checker finds any violations it returns a “counterexample”, which is a sequence of reachable states beginning with a valid state and ending with state that produces a violation. Model checking in automation testing context is used for two reasons [Gargantini, 1999]: first, the model checker can be used as an oracle for test outcomes evaluation and second, the model checker is able to generate counterexamples that are used to construct test sequences. This is the major challenge for testing based on model checking because it is necessary to force the model checker to construct a test sequence. [Gargantini, 1999] proposes a method to generate tests case from properties. In case we have to verify property P, we have to translate it in a temporary logic (e.g. CTL) and we have to give it to a model checker. However, our goal is to generate test cases, so we give the negation of P to the model checker. In order to demonstrate violation, the model checker produces a counterexample which is a trace of steps that represents a single test of a test suite.

2.1.1.3 Model-based system analysis using NuSMV

FBK will integrate in Crystal the extended version of the NuSMV model checker, namely a tool suite including the following tools: nuXmv, xSAP and OCRA. The suite supports the development and verification of complex, possibly safety-critical, embedded systems, covering different phases of system development. It implements a model-based approach for validation and verification, and supports several engineering activities, such as:

- Requirements validation:

  To check the quality (consistency, completeness) of a set of requirements
- Verification of functional correctness:

  To check the compliance of a system model with respect to a set of properties
- Safety analysis:

  To analyze the robustness of a system with respect to faults; it includes techniques such as Fault Tree Analysis and Failure Modes and Effects Analysis
- Contract-based architectural design:

  To drive the architectural decomposition of a system using contract-based design, and verify it using compositional verification techniques

The extended version of the NuSMV model checker will be developed by FBK, and adapted to Crystal needs, in order to support the IOS specification, and to integrate it in the Crystal RTP. New interfaces will be developed and integrated, according to users’ needs and as a consequence of requirements coming from the UCs. In particular, new formats will be defined in order to exchange verification data (e.g., traces and fault trees). Moreover, the following issues are of interest for NuSMV, and will be considered when during the design of the IOS: linking requirements and contracts with models, and supporting traceability of artifacts (e.g., tracing verification and safety artifacts to models).

The development will be carried out in a dedicated task in sub-project WP6.4, namely Task 6.4.6 (NuSMV brick development). For more details on the development and the capabilities of NuSMV to be integrated in Crystal, we refer to Deliverable D604.011.

2.1.2 Use Case coverage and application

This brick is indented as a consolidation brick. Specific use case needs will be identified in the next phase of the project

2.1.3 General improvement

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description:					

TBD	
Link to internal working documents:	

2.1.4 Integration and interoperability

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

2.2 B3.7 - Model-based requirements engineering

2.2.1 Description

Name:	Model-based requirements engineering
Contact:	Christian.webel@iese.fraunhofer.de
Dependencies	
License	
Additional information	

2.2.1.1 State of the art in model-based requirements engineering

2.2.1.1.1 Results from Systematic Mapping Study

In order to assess the state of the art in model-based requirements engineering for automotive systems, a systematic mapping study was performed. Particularly, we focused on approaches which model functional behavior and have been validated in industrial case studies.

We present in the following the research method, the presented modeling formalisms and which aspects can be modeled.

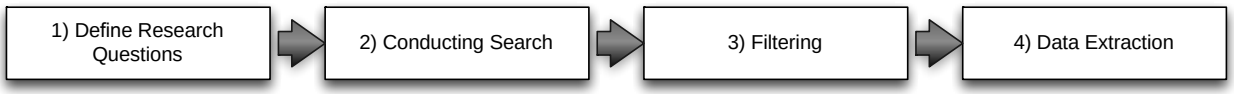


Figure 2-6: Research method

Figure 2-6 shows the employed research method. First, we defined the research questions to guide our research. In the second step, we conducted the search using keyword search on different literature databases. The resulting papers were filtered based on inclusion and exclusion criteria in step 3. We extract the data from the papers with respect to our research questions in the final step.

Research questions
RQ1: Which model-based requirements engineering approaches exist targeting automotive or embedded software?

RQ2: Which modeling languages are used in the different approaches?

RQ3: Which aspects of the software can be modeled by the different approaches?

RQ4. What is the level of industrial maturity?

Conducting search
The data collection was performed on IEEE Xplore and the ACM digital library (which also includes papers from SpringerLink, the digital library of Springer) with the following query:

(automotive OR embedded)
AND (intitle:requirement)
AND (model OR modeling OR formal OR executable)

The first part of the search string restricts the papers to those which include the term embedded and automotive as these are our domains of interest. The second part of the search string restricts the results to those which have requirement in the title. We restricted the search term ``requirement'' to appear in the paper's title since the term is used in many unrelated papers in the body text. Furthermore, it showed during definition of the search term that papers which deal with model-based requirements engineering used the term requirement in the title. Finally, we restricted the search to those papers which use the terms ``model'', ``modeling'', ``formal'' and ``executable'' in order to find only those papers which deal with model-based requirements engineering. The search resulted in 266 papers on IEEE Xplore and 298 papers in the ACM Digital Library.

Filtering
Based on the search results, we manually filtered each paper by inclusion and exclusion criteria. The inclusion criteria were: ``behavioral models, automotive or embedded systems focus, requirements engineering''. The exclusion criteria: ``focusing only on tracing, unrelated to model-driven development, focusing only on variability, focusing only on non-functional properties.''.

The filtering was done on the abstracts of the papers initially and additionally on the papers themselves in case of doubts and all papers which were finally included. As a result of the filtering, we found 40 relevant papers in the ACM digital library search results and 74 relevant papers in the IEEE Xplore results.

Data Extraction
All selected papers were classified according to different categories with respect to the research questions. As a pre-defined fixed set of categories was not sufficient to handle the diversity of modelling languages used as well as validation and verification activities, the list of categories was extended during the data extraction phase.

We define ``industrially relevant papers'' as those, which report about the application of an approach to an industrial system or to a standard like the European Train Control System (ETCS), which on the one hand has a considerable complexity and also will be implemented by companies.

Results
In the following, we present an overview of the results from the mapping study. We show the number of all relevant papers and additionally give references for those that report on industrial application as they are the most relevant to Crystal (See Figure 2-7, Figure 2-8, and Figure 2-9).

Domain	Papers reporting on industrial application
Automotive	[S20], [S1], [S27], [S3], [S18], [S16], [S28], [S15], [S10], [S2], [S26], [S21], [S9], [S11], [S17], [S23]
Avionics	[S25], [S14], [S13], [S12]
Health care	[S30], [S19]
Railway	[S6], [S8]
Smart card system	[S4]

Table 1. Papers reoprting on industrial application classified into domains

Figure 2-7: Papers reporting on industrial application

Modeling languages	All papers	Papers reporting on industrial application
State machines	22	[S26],[S13],[S27], [S12]
Sequence charts	14	[S11], [S1], [S2], [S9], [S19]
UML in general	10	[S16], [S19]
Petri nets	7	
SysML	5	[S3]
Other	55	[S1], [S30], [S15], [S10], [S17], [S14], [S4], [S6]

Table 2. Most popular modeling languages found in the papers

Figure 2-8: Most popular modelling languages

Modeling aspects	All papers	Papers reporting on industrial application
Components	11	[S19]
Refinement	9	[S26]
Requirements traceability	7	[S27]
Communication	5	[S11]
Discrete, event-based	34	[S1], [S28], [S13], [S19], [S6], [S12]
Time/timing	17	[S20], [S6]
Continuous/hybrid	10	[S6]

Table 3. Most popular modeling aspects found in the papers

Figure 2-9: Most popular modeling aspects

From our initial results, 20% were relevant with respect to our research questions but only 4% reported on experiences in an industrial setting. The majority of the papers were published in the last five years.

The papers, which report about industrial use cases, use mainly state machines and sequence charts for behavioral modeling with a clear focus on discrete, event-based systems. Interestingly, there is less support for continuous and hybrid modeling of requirements and timing constraints. However, automotive systems clearly include both continuous as well as discrete behavior and must satisfy hard real-time requirements.

Finally, a publication about an empirical assessment of a model-based requirements engineering approach in an industrial setting could not be identified, i.e., there is no evidence that the proposed methods and modeling languages for requirements engineering are better than existing requirements engineering techniques.

2.2.1.1.2 Model-based requirements engineering using Controlled Natural Language

Using Controlled Natural Language (CNL) for requirements specification is one possibility to represent requirements in a model-based, textual form. CNL is a subset of natural language that has a restricted grammar and vocabulary. It can be used to bridge the gap between natural language and formal languages. Boilerplates are a popular way of using CNL for requirements. Boilerplates are predefined sentence templates (fixed structure) that contain variable parts (placeholders) that have to be filled by the requirements engineer. These variable parts can be restricted by using only language elements of a glossary. Figure 2-10 shows an example for a boilerplate and a corresponding boilerplate based requirement. More information regarding boilerplates and processing boilerplate requirements can be found in [Hull, 2011]. Further boilerplate-based requirements languages are described in [Videira, 2005], [Denger, 2003] and [Daramola, 2012].

Boilerplate:
The <system> shall <function> every <quantity> <unit>.

Corresponding requirement:
The *HCU* shall *check SOC* every *1 ms*.

Figure 2-10: Example for boilerplate based requirement

The application of boilerplates for requirements engineering is a good instrument to improve the quality of requirements. As the requirements are specified consistently with defined terms, the ambiguity of requirements can be reduced. Further, the semi-formal notation of requirements makes it possible to automate some processing steps. Two possible approaches are described below.

Checking boilerplate based requirements

The prototype tool DODT, implemented in the CESAR project, has been used for early validations of boilerplate requirements. Using boilerplates in combination with a domain ontology enables checks for completeness, consistency, unambiguity and so on. Linking the domain ontology to attributes (placeholders) of boilerplates enables automated reasoning in order to perform these checks. More information regarding this approach can be found in [Farfeleder, 2011a] and [Farfeleder, 2011b].

In the CESAR use case, the semi-formal boilerplate requirements have been converted into formal pattern requirements [Reinkemeier, 2011] in order to perform some more detailed analyses on the requirements. A similar approach will be part of WP607.

Generating models from boilerplate based requirements

In [Holtmann, 2010] and [Holtmann, 2011] boilerplates are used to generate graphical models. Text-to-model and model-to-model transformation techniques enable the generation of SysML models from textual requirements. Depending on the used set of boilerplates, models like statecharts or block diagrams can be created.

For the generation of behavioral models like statecharts, activity diagrams or sequence diagrams the combination of use case based descriptions with boilerplates makes sense as well. Use cases are a well-known method for specifying the intended behavior of a system. Hence, a textual behavior description facilitates the generation of behavioral models as the sequence of activities is already described.

In [Yue, 2011] several transformation approaches between requirements and models are described. Among others, boilerplate based requirements are used for the transformation of textual requirements into graphical models.

2.2.1.1.3 Planned approach in Crystal

In Crystal, the boilerplate-based requirements should be used as a quality gateway to improve natural language requirements and for the generation of models. The aim is to find a language that covers both the possibility to specify different kinds of requirements and the generation of models. The application of the restricted language should be guided within a prototype implementation which should support the requirements engineer in easily applying the defined language. Moreover, the prototype should support the generation of model elements, such as SysML blocks or states, as well as diagrams.

2.2.1.2 State of the practice in model-based requirements engineering

In order to assess the state of practice in model-based requirements engineering for automotive systems, an interview study is currently being planned. The aim of the study is to add a detailed view on the state of practice to the information obtained in the online survey (described in section 2.1) and to the information obtained from the systematic mapping study (described in section 2.2.1.1). In particular, we want to identify reasons, which prevent practitioners from using models within requirements engineering, and existing solutions, in which models are used during requirements engineering. The study shall identify gaps in the current body of knowledge and motivate future research activities within CRYSTAL.

So far, the research questions were formulated and the study format was planned. Right now, we are approaching companies in order to schedule interviews for the study. We aim to start with two companies at different positions within the value chain (e.g. one OEM and one 1st Tier supplier). Within those two companies, we plan to perform at least four interviews covering at least two different roles (e.g. Project Managers and Requirements Engineers). Based on the results of these interviews, the interview study could then be extended to further interviewees, roles, or companies.

2.2.2 Use Case coverage and application

The brick is integrated with the Use Cases 3.1 and 3.4. In both use cases, we worked with real requirements documents to identify requirements for the model-based requirements engineering approach, particularly in our case requirements for extensions of Modal Sequence Diagrams. Modal Sequence Diagrams are an extension of UML sequence diagrams that supports the manual and automatic simulation of a set of sequence diagrams as well as the synthesis of an implementing state machine.

In Use Case 3.4, we worked on requirements for a draft document for the upcoming “Worldwide harmonized Light vehicles Test Procedures (WLTP)” standard that contains requirements how fuel consumption testing for vehicles has to be performed. The standard contains a variety of requirements addressing the human operator, mechanical system parts as well as software parts. From the latter, we particularly focused on the requirement for the shifting of gears. We developed a set of Modal Sequence Diagrams, which capture these requirements. In Use Case 3.1, we worked similarly to 3.4 on the Volvo demonstrator – an adaptive speed limit system.

The WLTP standard contains a variety of requirements addressing the human operator/vehicles (System under Test), the test equipment as well as software parts (See Figure 2-11 and Figure 2-12).

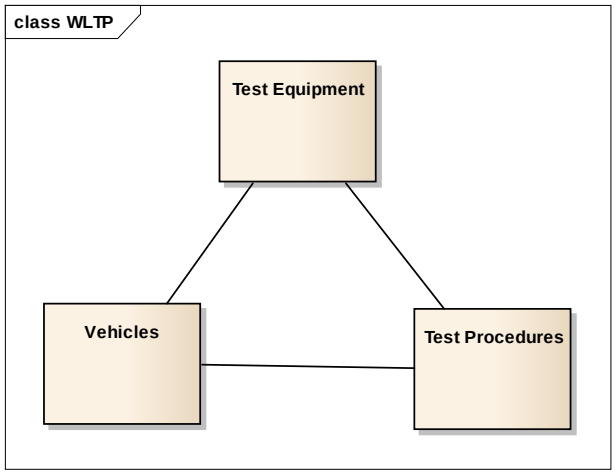


Figure 2-11: WLTP requirements overview

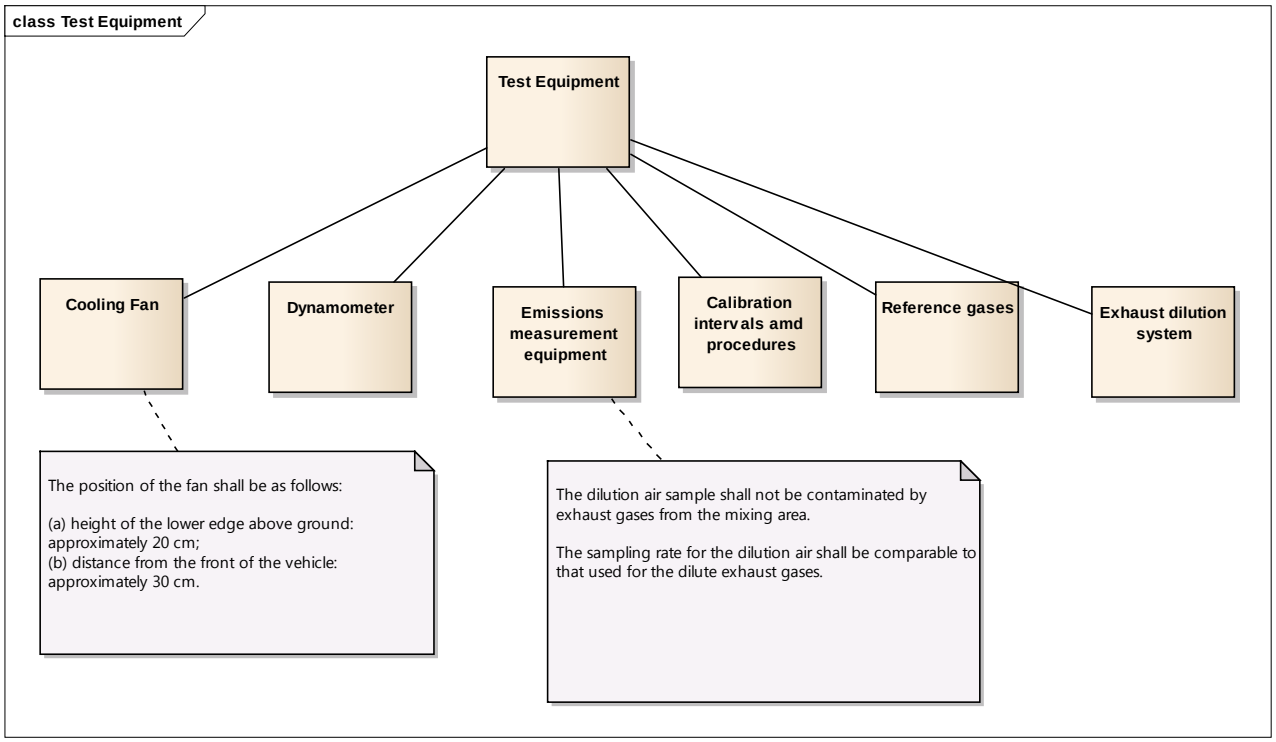


Figure 2-12: WLTP test equipment requirement kinds

In this brick, we focus on two things. First the structuring and formalization of the WLTP draft with focus on the elicitation and modeling of (legal and system) constraints. This includes the implementation of a UML/SysML profile in Artisan Studio (see brick ARTISAN studio). And second, we extend the work of CTH (Modal Sequence Diagrams for Annex II of WLTP) by state machines capture the gear shift requirements, to compare these two formal approaches.

In addition, in Use Case 3.4 the following activities are planned concerning boilerplate-based requirements:

1. The natural language requirements are transferred from HP Quality Center to the requirements semi-formalization tool.
2. The natural language requirements are semi-formalized with defined boilerplates.
3. The boilerplate-based requirements are transferred to HP Quality Center and linked with the natural language requirements.

4. The boilerplate-based requirements are transferred to the AVL V&V environment in order to automatically check the test results against the requirements.

From the results for the work in the use cases, we identified different additional requirements for Model Sequence Diagrams and the corresponding implementation in the eclipse-based ScenarioTools. First, there is a need to integrate continuous behavior into ScenarioTools as for example the gearshift requirements depend on whether the car is accelerating or decelerating. Second, time must be supported in different ways, e.g., clock and time guards as in Timed Automata are required for specifying timing requirements between gearshifts, as well as time periodic message exchange. Third, there is a need in the implementation to support validation activities like using Model Sequence Diagrams as test oracles or generating test cases for implementations. We are currently working on the corresponding implementations.

2.2.2.1 Identified requirements for extension

Regarding the formalization of the WLTP standard in general, and the modeling of Annex II (gear shift) with state machines in particular, one big challenge is how to model the algorithms and definitions of WLTP describing constraints, and how to trace and measure the impact e.g. on test equipment or procedures on change. A constraint is a requirement that is non-negotiable e.g. conformance to legal regulations or physical forces, and defines a hard boundary for a system. Second, also time has to be considered, as described before.

Further, following extensions have to be made concerning boilerplate based requirements:

- Definition of limit values within the requirement language (needed for checking the test results)
- Provide appropriate output format to support automatic checks of test results
- IOS concept for
    - exchanging requirements with requirements management tool
    - exchanging glossary terms

2.2.3 General improvement

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

2.2.4 Integration and interoperability

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

2.3 B4.7 - Guaranteeing real-time execution of critical features

2.3.1 Description

Name:	Guaranteeing real-time execution of critical features
Contact:	dominique.segers@barco.com
Dependencies	
License	
Additional information	

There is ever increasing product variability and the need to speed up time-to-market and reduce development time. This requires Barco to change their medical display platform from a hardware centric, custom platform towards a flexible software centric, Commercial-off-the-Shelf (COTS) platform.

This software centric medical display platform is confronting us with some challenges regarding real-time requirements, especially frame-rate guarantees.

Before a digital image reaches the eye of the viewer, a number of transformations have to be done in order for the digital data to be viewable by the viewer. For this case only the display image pipeline is considered: from an electrical input (typical DVI or Display Port) to pixels on the panel. Figure 2-13 shows a diagram that gives a high-level overview of a medical display image pipeline.

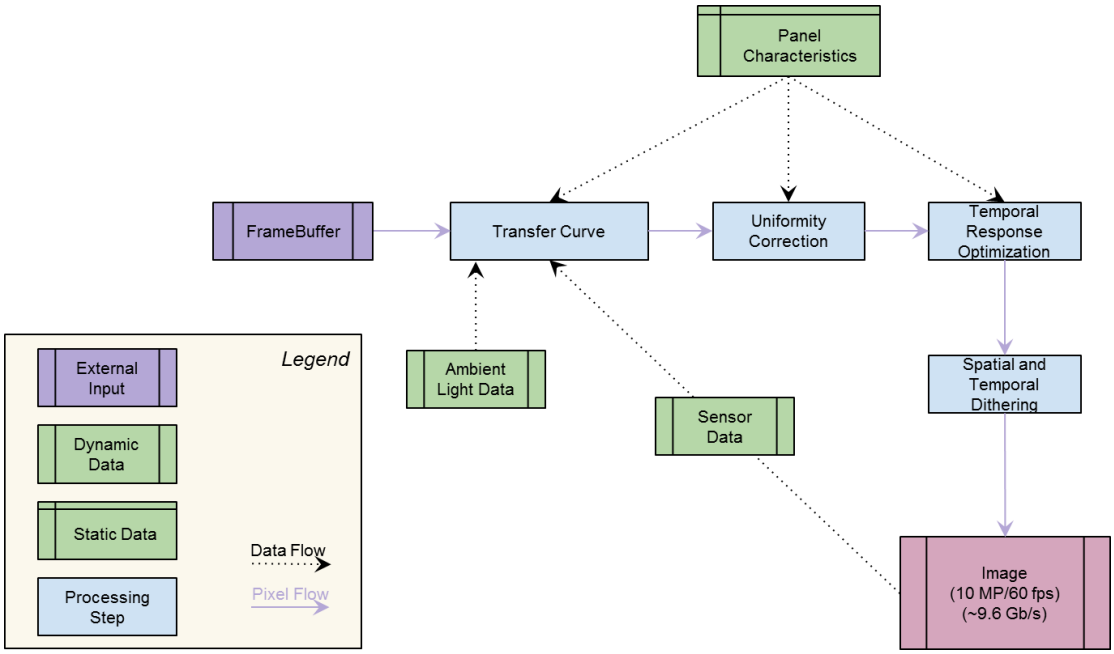


Figure 2-13: Typical Medical Display Image Pipeline.

Medical displays typically have a very high resolution, ranging from 3MP (Mega Pixels) to 10MP and higher, and a high pixel depth, 10-bit/channel or even higher. With a frame rate of 60Hz, this means a throughput of at least 18Gb/s. Guaranteeing this throughput in a software-based system is a major challenge.

2.3.1.1 State of the art in real-time guarantees

The imaging pipeline of current medical displays is developed using FPGAs. Guaranteeing real-time requirements is quite straightforward, as this is primarily a function of the FPGA size, clocks and available bandwidth.
On the other hand, guaranteeing these constraints in a software-based system running on COTS hardware is far from trivial. Those constraints not only depend on a real-time OS and the capabilities of the CPU, GPU, memory bandwidth, and availability, …, but also on the design of the software itself (parallelism, cache coherence, swapping, etc.) and the processes running alongside the own software (resource competition). Current practices often involve simplistic benchmarking and trial-and-error.

2.3.2 Use Case coverage and application

In order to be valid for a medical image for diagnosis, the displayed image must adhere to a number of legal requirements, such as no pixel loss, GDSF compliant, uniformity …
To this end, the digital input data is run through a number of transformations, the so-called medical image pipeline, thereby keeping the frame rate at 60Hz (important for displaying time-series of medical images).
Doing this in software is far more unpredictable than in hardware (FPGA).

By using model-based engineering as described in UC4.4 and UC4.5 a methodology will be introduced to design the software so that constraints, requirements and alternative architectures can be tackled and put under control very early in the design process.

Figure 2-14 describes the desired development process used for a software-centric and cost-effective replacement of the image pipeline. It is represented as series of activities (‘activity diagram’) that are executed to deliver the product.

The figure includes the legend for all symbols that are used to describe this process. The activities are represented by a rectangle. Each activity has a number of inputs and outputs that are represented by a rectangle with a snipped corner. The start of our development process is indicated by an open circle. The process ends with a completed product and is represented by a solid black circle.

The process is repetitive and uses an agile approach to come to the final product in series of refinement steps, the dotted box surrounding step 3 till 8 indicates one sprint cycle in the agile process.

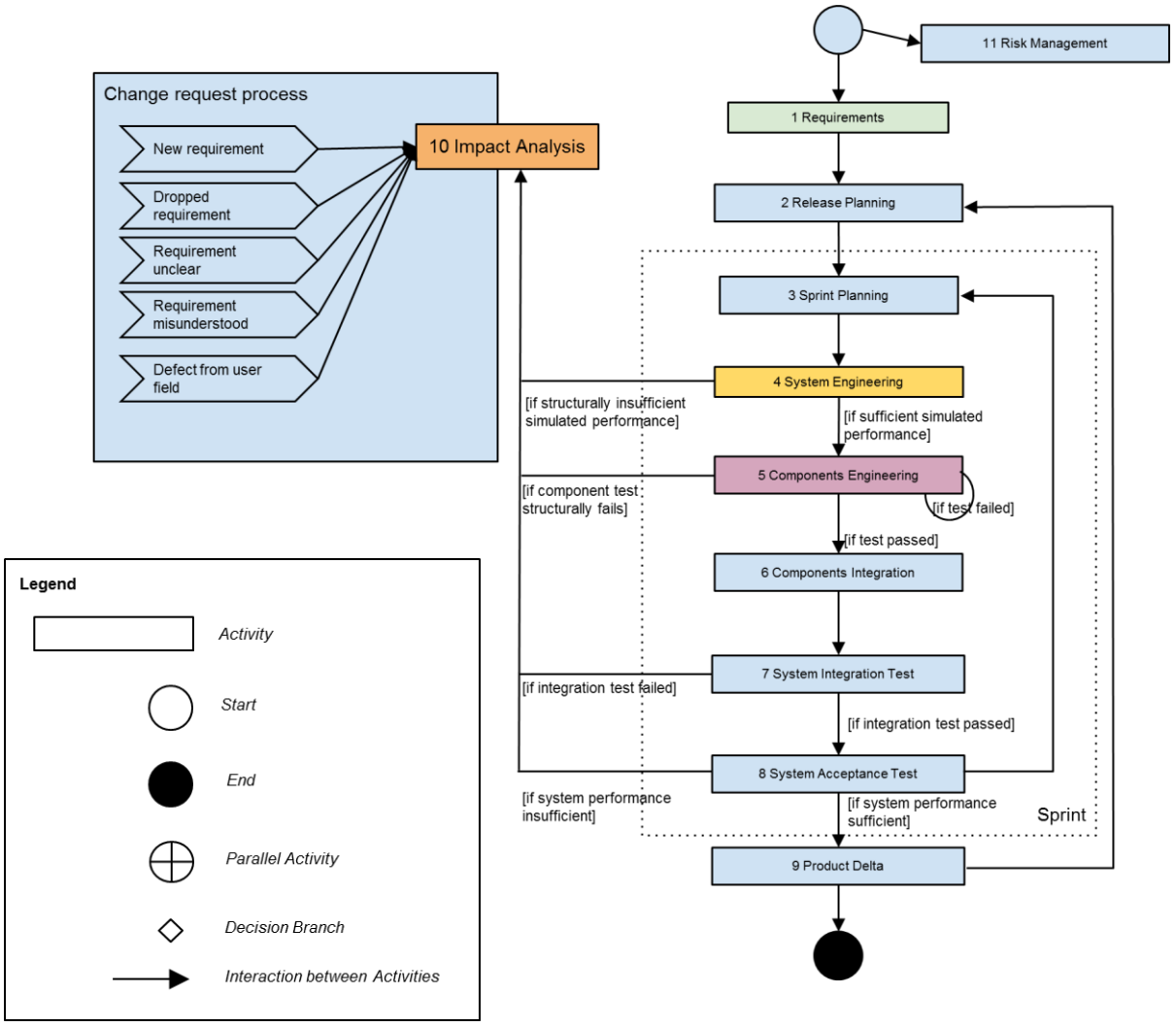


Figure 2-14: Software Display System Process

2.3.2.1 Requirements

The following requirements have been identified in the first planned CRYSTAL iteration (more to follow):

- Traceability of the requirements and constraints from a requirements tool into the modeling tools.
- Reusability of previous (successful) models.
- Continuous testability and versioning of the different models.
- Modeling of a virtual hardware platform (CPU, GPU, memory, concurrent processes).

2.3.3 General improvement

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD

Description: TBD	
Link to internal working documents:	

2.3.4 Integration and interoperability

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

3 Activities focusing on Tool Bricks

3.1 System design and analysis

This chapter runs under two major terms: System Design and System Analysis. The following selections describe tasks and activities performed under WP 6.3 which are listed here because they are considered to support the two important steps in product development.

System Design and System Analysis are two closely related items. Every design decision builds on former design decisions. The longer the list of unverified design decisions in sequence the higher the risk of having it dropped because of errors, later. Designs must be tested for various reason, each being one of customer requirements. They are not independent. System design is about resolving those interdependencies in a balanced way. The tighter the loop between design and design evaluation (analysis) the better the overall design. It is either containing fewer flaws or more features. Sometimes better design targets at better maintainability (modularity, extensibility or down-time), improved revenue or increased safety. Especially safety is considered to be difficult to provide without proper quality management and proper design.

In general, it can be observed that system design tools become more powerful. They support the engineers and designers not only on a specific level of abstraction but also provide support design of lower abstractions where needed. We can call this “deep architectures”. Despite the higher expressive power of deep schemes the complexity for humans remains the same and hence the chance of error is the same. At one hand engineers have to deal with fewer technical details under the hood but on the other hand the number of features and requirements at various technical levels increases. The increasing diversity of requirements is characteristic for modern and future engineering. This forces design tools to support engineers in keeping track of all the design constraints and to develop “semi-intelligence”.

Semi-intelligence is when tools guide engineers during the design process by inferring propositions about next steps. Systems making useful propositions and which simplify their execution a great deal can be used to strengthen engineering best-practice on the grounds of what is easiest to do is also best. Such semi-intelligence is heavily relying on well-connected data structures and project artifacts. Without this additional information it is hard to imagine how project specific support of that kind could be realized by the tools. Semi-intelligence is not as strong as expert systems design. It does not derive a full solution. However, semi-intelligent development environments would significantly lower future investment into expert systems and would give an edge over the competition, soon.

The strategy to connect artifacts and logical project items into a “big semantic map” via OSLC is exactly what CRYSTAL is trying to do. It will improve the overview over a broader landscape of requirements and WP 6.3 will try to foster a stronger use of this information in all design process steps.

Deep architectures – deemed very powerful scheme of expression – are also susceptible to subtle errors which come from the implicit interaction rules between the abstraction layers or the formalisms in use. Sometimes issues of deficits are simply matters of tool-specific implementation. In order to catch such glitches design must be analyzed at regular rates. The earlier and the faster the analysis can be executed and the broader the choices are the smaller is the necessary design increment and the risk arising from it. Therefore it is observed that analysis tools a) become integrated with design tools (eventually merging to the point of becoming indistinguishable) and b) the variety of analysis tool is increasing in order to keep pace with the diversity of requirements. This helps to build safer machines because safety requirements are very broad by their nature. With a strong interoperability layer (as OLSC is thought to be) WP 6.3 can hope to increase the diversity of tightly integrated analysis tools in any project.

3.1.1 B3.28 - Sparx Enterprise Architect

3.1.1.1 Description

Name:	Sparx Enterprise Architect
Contact:	Aleksander.Lodwich@itk-engineering.de

Dependencies	
License	
Additional information	

UML is a standardized graphical language for expressing structural and temporal properties of a (mostly computer-based) system. It fills a gap between the very generic graphs known from mathematics and computer science and the very details of a solution in a computer system implementation (code & configuration & distribution details).

By historical fact, UML was first designed to model the relationships between “objects” in relational databases, “objects” in computer software driving them and “objects” in real environment which are often human operators. It was observed that software architects sketch their ideas with some common symbolism and this has become the first set of UML diagrams.

The collection of elements in UML clearly reveals its technical nature and application niche. The design process for the various aspects of software requires fixating relevant technicalities in pictures, a feature that was nowhere defined before for abstract concepts like the ones used in software.

With UML the modeling of software has drawn equal with other engineering disciplines where the simplified definition of the so developed system has allowed increasing the overall complexity of produced systems. Especially, UML allows to concentrate on capturing important aspects of the final design and to start with the implementation of a complex system in a goal-oriented way. The success of UML has led to the introduction of enhancements which help to model aspects of embedded systems (SysML). In general, “visual design” has proved to be very convenient for humans. Development teams employing such technologies are observed to work faster to commit to less risk of error.

UML and many other graphical modeling schemas are a genuine pencil and paper technologies for the concept development phase. However, as products are maintained or improved the nature of the original drawings become repetitively relevant for the more recent design decisions. From such re-use of diagrams it has become more and more important to keep design and the implementation synchronized. As the elements in the diagrams “interact” witch each other, navigation between the different design views (partial models) has also become important. This is the point where on the one hand UML has grown beyond the concept phase and on the other hand where a smart development environment was urgently needed in order to accomplish the synchronization and navigation.

The Enterprise Architect (EA) from Sparx Systems is such a smart environment and has originally started as a modeling tool for commercial development environments. Through the wide success of UML and its broad application EA has grown to support development beyond mere UML modeling. Similar to Borland’s Together or IBM Rational’s Rhapsody, it can generate boilerplate code, definitions and API from the graphical model and helps to keep them synchronized during the complete life-cycle of the software. In fact, EA supports a broad range of techniques and methods observed during a complete life-cycle. It will capture requirements or support testing.

EA is an UML modeler. Models are considered good for engineering. Companies start to learn that drawing a solution instead of writing it up is not necessarily or automatically raising product quality, reducing cost or improving project success. That’s because there are two kinds of models that solve two different kinds of problems. Improvements of quality, a better return on investment or a reduction of project risk only come about when the right kind of models are used for the right thing. Tool chains must address this and guide developers to make the right decisions.

The first kind of models is the “simplified domain representation”. This class of models is used to express the essence of a setup. Such models adhere to natural concepts of human understanding and are typically much more compact. As a consequence these models reduce the number of possibilities to introduce errors and much better exploit available bandwidths in HMI communication. A tool allocated in this area is e.g. Simulink. Simulink is an example of a graphical modeling technique but non-graphical models are also very popular (think of algebra or custom 4G languages). Such models still have a 1:1 correspondence to what is modeled but the expressions can be quickly manipulated and then re-evaluated. Obviously, the natural habitat of such models is the concept phase or research phase where models of this kind stem from.

These models help to make faster cycles of statement and response and often enough new design patterns emerge to experts which tend to make the whole design process more efficient. Therefore, the quick discovery of a working solution is crucially depending on the right choice of the system of expression. The option to mix these systems seems advantageous. Modelica takes a radical approach here.

The other kind of models is “meta knowledge” which describes “templates” to a solution. UML belongs to this class of models. Such models describe families of solutions and how they develop and act over time.

By the very nature of embedded systems (fixed purpose, fixed environment, fixed resources, etc.) this is obviously not a hot topic and modeling them with UML is cumbersome. In PC systems which are very versatile, where solutions are generic, where communication units are complex and where time matters only to the point of order, it is very important to provide descriptions about the principles of systems instead their exact structure (every program instance might be very different in terms of used resources or internal configuration). For this kind of models it is important to prove that the structural grammar will not create illegal constellations or break assumptions.

However, as embedded systems become more networked, as they are equipped with more resources and as they start employing principles from mainframe computing (concurrency, filesystems, plugins, drivers, apps, etc.) a bridge is built between the two domains that establishes for an unusual flow of requirements into both directions. In one direction we observe an inflow of requirements asking for structural versatility and in the other direction we see an inflow of safety requirements. This combines to a new kind of question: How can we assure that a family of configurations is safe? Temporarily this is difficult to prove.

To answer this, we’ll need both kinds of models combined. The boundaries between embedded systems and general computing are ever more blurred. The mixing of the two types of models becomes also more common and tool-chains should provide transparent handling of them in one frame of context. For industrial purposes, the gap between the two has been narrowed with SysML. SysML was designed with respect to the needs of embedded development to model timely behavior more pronouncedly.

From this natural flow of considerations we observe three lines of further engineering improvements applicable to EA:

a. As more and more aspects of software development are found to be usefully modeled with graphical means, modeling tools grow into complete ALM solutions. Some development suites are found to be versatile enough in order to start designing non-technical systems or hybrid systems. This line of improvement will require open interoperability in order to handle the diversity of development environments.

b. Modern tools auto-code solutions from graphical descriptions. This makes development faster! In order to support this well, more and more technical details can be annotated with the steadily growing expressiveness of graphical languages. As manual extensions to auto-generated code become less and less common, graphical descriptions slowly lose their representative or informative character and turn into the actual code which must versioned, managed and evaluated. Therefore, tools must provide stronger support for a life-cycle of graphical code. EA is already strong here but cannot model own model variability.

c. Proving correctness of design is still manual work. Although static code analyzers and correctness provers exist, it seems that they only work under many design constraints. They often require an intermediate step of code production which becomes less and less of immediate interest. This is where we see tools like ASD:Suite come in, trying to bring together the strength of models with the strength of functional correctness proves. Since many companies already invested into UML models (and many did so by buying EA) they would like to harness tools like ASD:Suite in parallel. This implies some additional means of exchange or general interoperability. We would like to subsume developments in this area under the term “Static Model Analysis (SMA)”.

3.1.1.2 Use Case coverage and application

We provide service to all use-cases and consulting to all work-packages in SP6 regarding the application and interoperability improvements of EA. We have already observed that EA is used in non-conventional constellations where models of models or tracking of templates for models would be desirable. We have various weak confirmations that we will deal with EA in all three directions a, b & c.

3.1.1.3 General improvement

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

3.1.1.4 Integration and interoperability

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

3.1.2 B3.71 - AVL Cruise/Boost

3.1.2.1 Description

Name:	AVL Cruise/Boost
Contact:	andrea.leitner@v2c2.at
Dependencies	
License	
Additional information	

AVL Cruise is a powerful, robust and adaptable tool for vehicle system and driveline analysis based on simulations. It supports everyday tasks in vehicle system and driveline analysis throughout different development phases. This means that it supports driveline simulations in an early development stage as well as the step-by-step deployment of physical parts (e.g. an engine may be coupled with the driveline simulation and tested on a test-bed).

BOOST is an advanced and fully integrated "Virtual Engine Simulation Tool" with advanced models for accurately predicting engine performance, acoustics and the effectiveness of exhaust gas after treatment devices. It supports engine development such that for a given vehicle concept, the required torque and power can be delivered in combination with optimized emissions, fuel consumption and passenger comfort (acoustics and transient behavior).

AVL Solution

BOOST provides an engine simulation tool applicable from the concept phase up to ECU calibration, addressing the needs of engine and powertrain simulation projects. It is applicable for the analysis of individual components and all over systems simulation, with various modeling depth levels. It can easily be

linked with CRUISE (Vehicle System and Driveline Analysis tool) for Vehicle Thermal Management System optimization.
The highly flexible structure of CRUISE enables changing an existing driveline within minutes. Adding hybrid components to a conventional vehicle can be done with a few mouse clicks, using electrical components designed for HEVs. CRUISE can be easily linked with other simulation tools for sub-system integration of vehicle thermal management systems, vehicle control systems, driving dynamics and handling tools, components and sub-system test rigs and HiL systems.

Starting with only a few inputs in the early phases, the model matures during the development process according to the continuously increasing simulation needs.
AVL CRUISE is typically used in powertrain and engine development to optimize the vehicle system including cars, busses, trucks and hybrid vehicles, its components and control strategies with regard to:

- Fuel consumption and emissions for any driving cycle or profile.
- Driving performance for acceleration, hill climbing, traction forces, and braking.

AVL CRUISE is also used for tasks like:

- Evaluation of new vehicle concepts such as hybrid powertrain systems (e.g. regarding their performance).
- Simulation and analysis of standard and new gear box layouts like DCT and AMT.
- Analysis of torsional vibrations of elastic drivelines (under dynamic load).
- Drive quality assessment of transient events such as gear shifting and launching.
- Simulation and analysis of vehicle thermal management.
- Energy flow analysis, analysis of power splits and losses within components.

3.1.2.2 Use Case coverage and application

AVL Cruise will mainly be used in UC3.4. The main purpose of the tool is an early validation of the powertrain for vehicle simulations in an office environment as well as in an engine test bed environment.

UC 3.4 has defined the following requirements for this tool:

- Linking requirements (e.g. HP Quality Center) to AVL Cruise model elements.
- Integration of security aspects (e.g. who is allowed to change data).
- Reuse of simulation and calibration data from office to test bed phase.
- Development frontloading through early vehicle simulation and calibration iterations.

One main part of the integration is the support of traceability between different aspects (e.g. requirements, security) and CRUISE model elements as illustrated in Figure 3-1. In the UC, the requirements will mainly be stored in different representation and formalization levels (in tools such as HP Quality Center, PTC Integrity, Artisan Studio using SysML models, Boilerplate tools, etc.) and should be linked to the respective technical realization as for example CRUISE model elements. The interface should be open in order to be applicable to different requirement management tools, but will be demonstrated using one of the aforementioned tools. OSLC seems to be a suitable concept for the technical realization. The evaluation of the applicability of OSLC and the implementation of the interface are an essential part of this task. Another aspect which is of importance is the assignment of security information to model elements, to ensure that only authorized persons are able to change them.

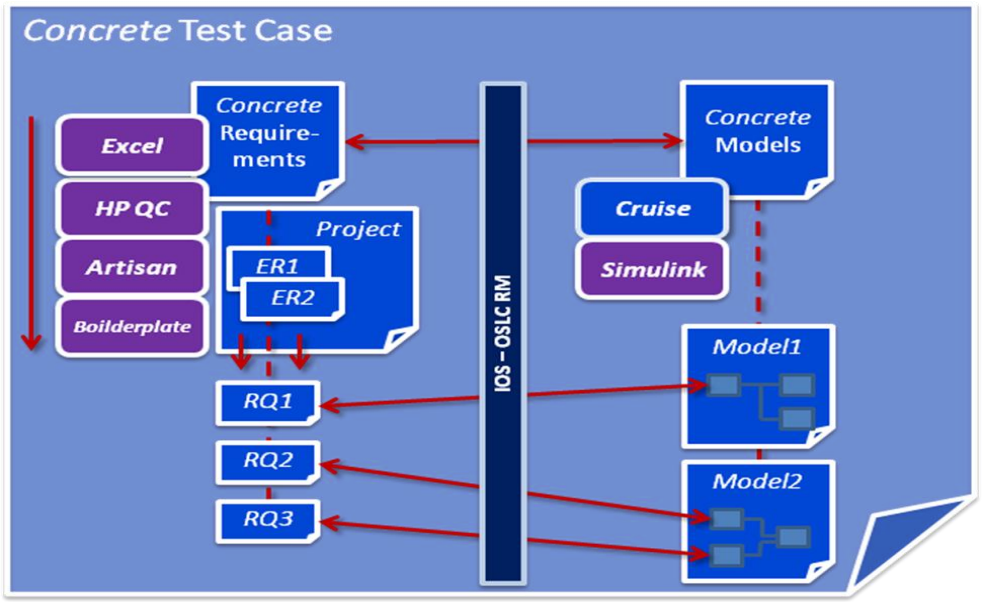


Figure 3-1: Linking requirements to CRUISE model elements

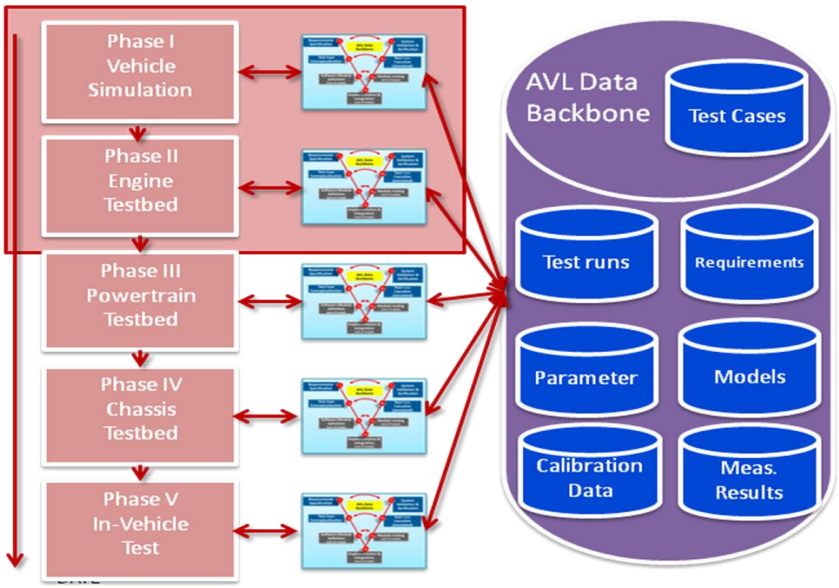


Figure 3-2: Reuse of simulation and calibration data from office to test bed phase

One core target in UC3.4 is the re-use of data across different development phases, whereas each phase is based on its own tools, processes and data representations. The focus in this project will mainly be on testing phase I, vehicle simulation, and phase II, engine testbed, as shown in Figure 3-2. The main difference between these two development stages is the substitution of the engine model by a real engine. This engine is interacting with the simulation model representing the drivetrain. The engine is operated on a test bed and gets stimuli from the model and feeds back output signals.

If simulation is combined with calibration in an early vehicle development phase, the design space exploration is enriched by additional possibilities: An advantage of this approach is for instance the possibility to include the evaluation of an optimized powertrain design in the calibration process, which (due to the fact that it is built in hardware later on) could not be done easily in a later development phase. Figure 3-3 gives an overview of the affected tools and data as well as of the process of calibration iterations.

Within one iteration the results of a first simulation can be used as an input for an optimization and calibration tool (such as AVL Cameo). Given some target values, AVL Cameo calculates an optimized driveline calibration based on a calibration model. This calibration is evaluated by a simulation in the next iteration round. Throughout several iterations the design can be adapted and optimized already in a virtual environment.

Currently, one main drawback is the fact that calibration and parameterization data cannot be reused straightforward in the subsequent development stages.

The realization of an AVL data backbone is intended to improve the reuse of simulation models, calibration data (applied also on these models), and so on throughout the different phases of vehicle development. CRUISE is one of the central vehicle simulation tools and therefore has to exchange information with different other tools and databases, respectively. One key focus here will be the tight integration of CRUISE with calibration tools such as AVL Creta/Cameo (see WP T6.10.7) in order to calibrate the driveline model. These tools need to exchange calibration data (Cameo → Cruise) and simulation results (Cruise → Cameo) via a defined interface. Since the data should not only be available in these tools, but also in other development stages with their respective tools, it could be stored in a central database. All tools which need to access the data would need to have an interface to this common database. Another possible solution would be to use the concept of Linked Data to ensure traceability between different data representations or a combination of both approaches.

The data backbone is part of WP613 and therefore described in more detail in the respective deliverable. The integration of AVL CRUISE and this data backbone will be implemented in this task, but with a close collaboration with WP613.

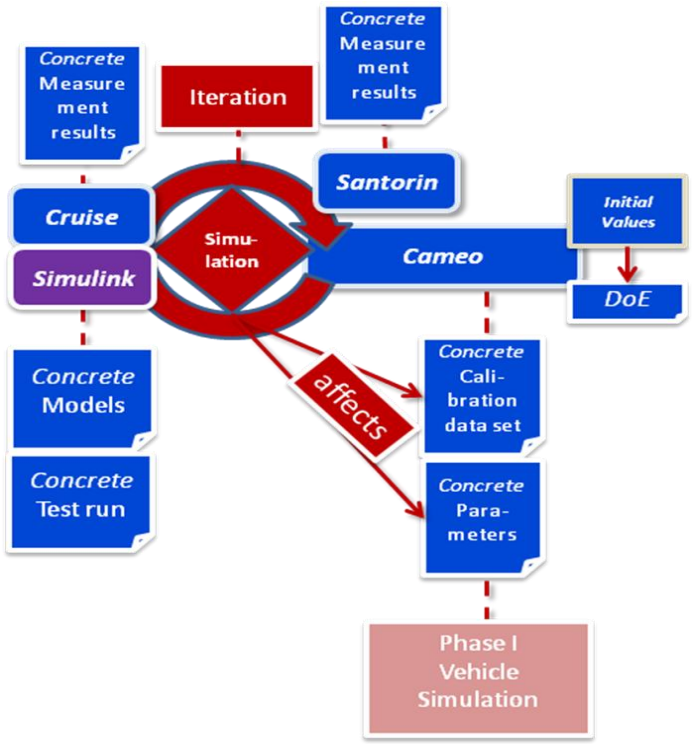


Figure 3-3: Development Frontloading through early vehicle simulation and calibration iterations

3.1.2.3 General improvement

The definition of Technical Items will be part of the next phase of the project.

TI NAME:

TI_ID	CRYSTAL_TI_x	Kind of TI	T, M, MM, G	Contact email	TBD
Description: TBD					
Link to internal working documents:					

3.1.2.4 Integration and interoperability

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

3.1.3 B3.65 - Rubus ICE

3.1.3.1 Description

Name:	Rubus ICE
Contact:	kurt.lundback@arcticus-systems.com
Dependencies	
License	
Additional information	

Arcticus offers Rubus Integrated Component Environment (Rubus-ICE) for component modeling, pre- and post-runtime analysis and software synthesis. The IDE consists of a set of tools for design, analysis, and synthesis of component-based real-time systems based on Rubus Component Model and is plugin-based and possible to extend to other models and run-time frameworks.

Rubus-ICE and Rubus-RTOS products are in use and deployed in a number of successful projects by our customers.

3.1.3.2 Use Case coverage and application

Adapt analysis methods and Rubus ICE to the actual needs of Software Engineering by enabling the seamless combination of functional constraints and requirements with target environment such as resources. Thus giving the software engineer one holistic analysis framework.

Arcticus participation will be based on the latest EAST-ADL specification including TADL2 timing model aspects.

In this project Arcticus will adapt and extend Rubus-ICE to support EAST-ADL component model and explore the possible pre-runtime timing analysis that can be performed given the available information from EAST-ADL. The precision on a high-level model i.e. EAST-ADL timing-analysis is expected to differ from the

more exact timing-analysis performed in the tool today since the information is not available in a high level system model.

The IOS integration will be implemented by reading and writing the standard EAXML file-format.

Based on the existing Rubus models and tools the EAST-ADL and the TADL2 models are integrated into the tool chain. Currently we have a prototype of a graphic editor for these models. We want to investigate, design and implement model transformation bi-directional and to improve our analysis model to adapt to these models.

3.1.3.3 General improvement

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

3.1.3.4 Integration and interoperability

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

3.1.4 DTFSim – Data Time Flow Simulator

3.1.4.1 Description

Name:	DTFSim – Data Time Flow Simulator
Contact:	Alexander.Hanzlik.fl@ait.ac.at
Dependencies	
License	
Additional information	

The DTFSim is a discrete-event simulation environment which focuses on design and analysis of the network architecture of electronic control systems. For this purpose, so-called **event chains** from sensors to actuators are modeled and simulated.

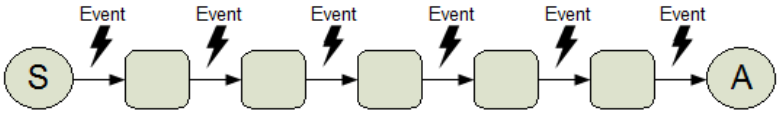


Figure 3-4: Event chain for timing analysis

Event chains are sequences of linked **base elements**, originating from sensors and ending up at actuators. These base elements are provided by a modular assembly system.

Typical base elements are

- **Sensor** (model input),
- **Actuator** (model output),
- **Processor** (freely programmable, for dedicated functionalities like software component functions),
- **Controller, Transfer, Line, Extractor** (for modeling communication buses)

Each base element has the following properties:

- It has one input, one output and a propagation delay.
- It provides a dedicated functionality which relates to the transformation of input values to output values.
- The output of a base element is the input of the next base element in the event chain.
- It is triggered by events. When a base element is triggered, it processes its input, generates an output value according to its functionality, writes this value to its output and generates an event for the next base element in the event chain.

Doing so, events and data propagate over time along event chains.

The DTFSim provides a template mechanism, so-called **super elements**. Super elements are constituted from base elements and are used to build up complex structures like ECUs or communication buses. Once created and stored, they can be used in different simulation models.

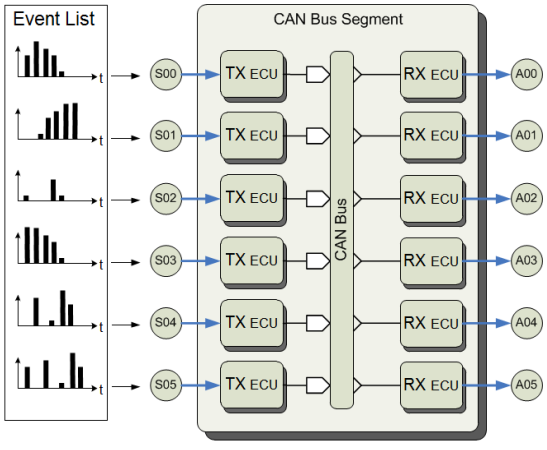


Figure 3-5: DTFSim model of a CAN network segment

For the modeling of communication networks which contain of one or more network segment(s), the DTFSim uses four types of super-elements:

- **TX ECU**s which process sensor inputs and transmit frames on the **Bus**,

- **Bus**es (CAN, FlexRay or Ethernet) which are responsible for message arbitration and frame transmission,
- **RX ECU**s which receive frames from the **Bus** and process actuator outputs,
- **Gateway**s which connect different network segments.

The sensor inputs of the DTFSim models are triggered by so-called **event lists** which describe typical scenarios over time, e.g. the different positions of the brake pedal during a drive cycle. All events observed during simulation are stored and used for subsequent analysis of the simulation results.

A typical DTFSim workflow comprises the following steps:

- **Configuration**
  The following information is needed for creation of a DTFSim network model:
  - **System Architecture**
    ECUs and how they are connected
  - **Network Parameters**
    Communication protocol (CAN, FlexRay, Ethernet, …) used for communication between the ECUs; bandwidth settings; message catalogue (messages sizes, message priorities, message send periods, communication cycle duration)
  - **Component Parameters**
    Component functions (the input-output transformation functions for the base elements, like software components); component latencies (the propagation delays of the base elements, like the execution times of software components)
  - **Timing Requirements**
    The maximum signal propagation times from sensors to actuators.

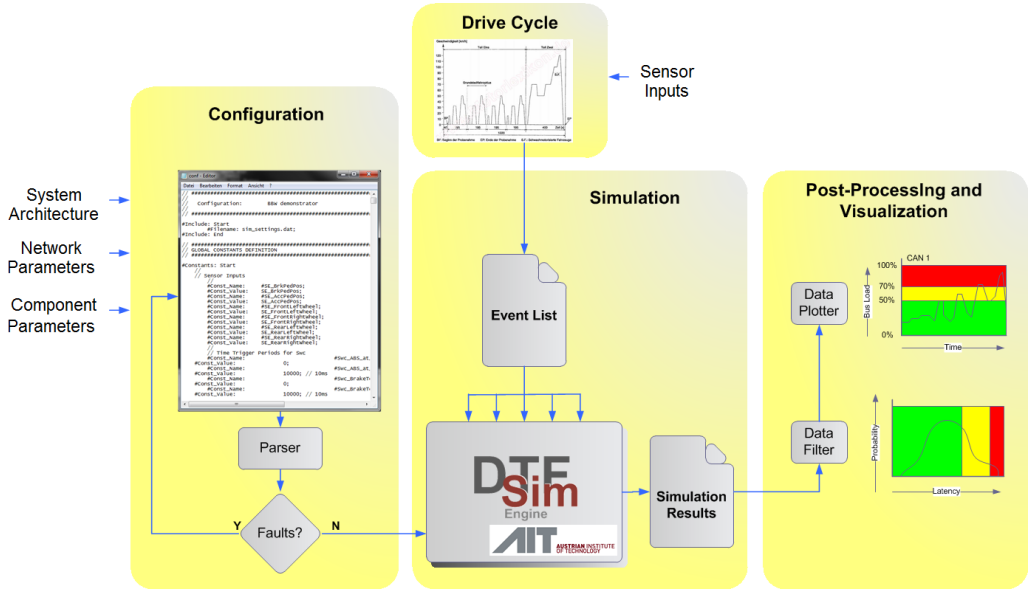


Figure 3-6: DTFSim design and analysis workflow

- **Drive Cycle (Event list) generation**
  The creation of event lists for stimulation of the sensor inputs.
- **Simulation**
  Model execution according to the **Configuration** and the **Drive Cycle**.

- **Post-Processing and Visualization**
  Analysis and visualization of the simulation results. In this step, the **timing analysis** takes place.

3.1.4.2 Use Case coverage and application

The DTFSim supports the VOLVO Use Case UC3.1. The two thematic priorities of the DTFSim application in UC3.1 are

- Performance analysis of the communication network
- Timing analysis of time-critical event chains

According to the DTFSim workflow description, the following data will be needed for simulation of the system:

- Functional components and system topology (ECUs and their interconnection)
- End-to-end latency requirements on functional components (like WCETs of software components)
- Timing requirements (like maximum signal propagation times from sensors to actuators)

The ongoing integration into MBAT RTP will be updated to the CRYSTAL RTP and extended by integration timing analysis tools defined by the use case. The requirements engineering and architecture model integration already worked at in MBAT will be improved based on user feedback.

3.1.4.3 General improvement

AIT will improve and extend the brick DTFsim in task 6.3.10. The following improvements and extensions are scheduled:

The tool shall be integrated with other timing analysis tools, thereby achieving a holistic timing analysis for a system. Additionally, usability shall be improved by providing a graphical user interface, matching the requirements of the use case partner Volvo. The main goal is to reach a maturity level of the integrated tool fit for day to day application in industrial use.

The predefined network libraries (currently CAN and FlexRay) will be extended by Ethernet. In addition to the existing reporting mode, a GUI for interactive analysis and result visualization will be added.

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

3.1.4.4 Integration and interoperability

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					

Link to internal working documents:	

3.1.5 B3.79 - ARTISAN Studio (Task 6.3.15 – FhG)

3.1.5.1 Description

Name:	ARTISAN Studio
Contact:	Christian.webel@iese.fraunhofer.de
Dependencies	
License	
Additional information	

Fraunhofer IESE will implement its SysML profile for WLTP from Task 6.3.3 in Artisan Real-time Studio (RTS). Further, we will support the integration of RTS into the CRYSTAL RTP along the corresponding IOS. There are currently ongoing discussions with Artisan regarding the RTP integration with OLSC.

3.1.5.2 Use Case coverage and application

The definition of use case needs will be part of the next phase of the project.

3.1.5.3 General improvement

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

3.1.5.4 Integration and interoperability

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

3.1.6 B4.14 - Functional and performance analysis

3.1.6.1 Description

Name:	Functional and Performance Analysis
Contact:	elluna@iti.es, rjuan@iti.es
Dependencies	
License	
Additional information	

Functional & Performance analysis tool provide features for analyzing the performance and functional requirements at early stages of development. The analysis is based on timing and resource allocation for evaluating the goodness and suitability of the proposed design. Therefore, this tool is centered in the evaluation of the system behavior of the product.

3.1.6.2 Use Case coverage and application

This tool will be used in the use cases UC405 and UC406 of the healthcare domain.

- UC405 - SW centric scalable safety critical medical display platform

  The use case provides the development of a medical display that as new feature moves current functionality from hardware (FPGA) to a software centric platform based on GPUs. This new design will give more flexibility and eases the inclusion of new functionalities, but it introduces new challenges in development and design.

  A performance analysis tool can be used in the design phase to help in the selection of the right architecture to fulfill all the requirements

- UC406 - An intelligent infusion controller for Blood Pressure regulation in Operating Room

  The goal of this use case is to incorporate tools to support the development process of an intelligent infusion controller for facilitating certification processes.

  The product to be obtained in this development process is a system that operates delivering vasoactive drugs with the ultimate goal of reducing patient´s hypertension, and precisely controlling blood pressure measurements in a patient undergoing surgical intervention in Operating Room or in post cardiac surgery in ICU (intensive care unit).

  In this case the performance analysis tool is used in the design phase to help in the selection of the right architecture to fulfill all the requirements.

The UC405 and UC406 add the following requirements to the Performance Analysis tool:

- UC405/UC406: The tool shall provide a graphic modeler that supports rapid and assisted modeling of the system.
- UC405: The tool shall support GPU architecture/behavior.
- UC405/UC406: Performance checks shall be based on timing and resource allocation.
- UC406: The tool shall detect functional and performance issues related to timing and resource allocation, in order to avoid situations under which the obtained product does not guarantee the established requirements.

- UC405/UC406: The tool shall support the mapping of modeled components with the product requirements. In this way the solution supports the identification of non-fulfilled requirements based on the functional and performance issues detected.
- UC405/UC406: The tool shall generate reports with evaluation results. The report must identify the detected functional and performance issues and possible potential issues.
- UC405/UC406: The tool shall be based on well-established languages and technologies in order to avoid dependencies on obsolete and/or abandoned solutions that jeopardize its use and maintenance.
- UC405/UC406: The tool shall be easy maintainable and provide a long-time period of maintenance.
- UC405/UC406: The tool shall run, at least, on Windows and Linux systems.

The UC405 IOS requirements are related to improve the architecture design by analyzing the performance of the system architecture. UC406 IOS requirements are related to improve the requirements traceability during the project, in order to ensure that all them are correctly covered and handled in the design stage

The following list details the currently identified requirements for both use cases:

- UC405: Basic IOS requirement: The tool shall be able to import requirements from a requirement management tool. In this way the proposed brick must be able to connect to a requirement management tool for importing the requirements relevant for the architecture design. The result is an architecture design and therefore, the brick should provide functionality for linking the imported requirement with the components in the architecture. This mapping must to perform a follow up of the requirements fulfillment from a functional and performance point of view.
- UC406: Basic IOS requirement. The tool shall be able to import requirements from a requirement management tool. In this way the proposed brick must be able to connect to a requirement management tool for importing the current list of requirements. Moreover, the brick should provide functionality for linking the imported requirement with the components in the model in charge of satisfying them. This mapping must to perform a follow up of the requirements fulfillment from a functional and performance point of view.
- UC405/UC406: As advanced IOS requirement the tool shall be able to generate status information about the fulfillment of each functional and performance requirement and functionality for including this information in the Requirements Management tool.

3.1.6.2.1 Integration into the Use Cases

This brick will be used in the “Architectural Design” phase of the development process of UC405. This phase is done once a first Requirement Specification has been achieved, and before starting the “Detailed Design” phase.

This brick will also be used in the “Rapid prototyping of architecture and design” phase of the V process model of UC406. This phase is done once a first Requirement Specification has been achieved, and before starting the “Modular Decomposition” phase.

3.1.6.3 General improvement

During the CRYSTAL project it will be implemented and provided the basic tool functionality and the IOS basic required support.

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description:					

TBD	
Link to internal working documents:	

3.1.6.4 Integration and interoperability

Regarding tools integration, from IOS Requirements it is derived that this brick must be able to interact through OSLC with a Requirements Management tool. Therefore, support for OSLC Core and Requirements Management specification is needed.

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

3.2 Architecture analysis and exploration

3.2.1 B3.70 - ASD:Suite

Note: *Verum has stepped out of the project right after this first phase of the project. Therefore, there will be no update of this brick in future versions of this deliverable.*

3.2.1.1 Description

Name:	ASD Suite
Contact:	jos.langen@verum.com
Dependencies	
License	
Additional information	

The ASD:Suite is a software design platform based upon Verum's patented Analytical Software Design (ASD) technology. ASD makes it possible to create systems from mathematically verified components.

The ASD:Suite is used to define and (automatically) verify models, and to (automatically) generate fully executable source code from these models. The (discrete event based) models specify both structure and behavior of services, and of components that implement and use these services.

3.2.1.2 Use Case coverage and application

The objectives for the ASD:Suite in the light of the Crystal project can be summarized as:

- Interoperability of the ASD:Suite with other tools
- Improvements and/or extensions to the ASD:Suite supporting architecture analysis in terms of a/o structure, complexity and functional correctness
- Improvements and/or extensions to the ASD:Suite supporting *validation* of components, next to the existing automatic *verification* of components.

Next to WP603, Verum is also involved in WP301 that considers the Volvo use case. Within this use case, the focus is on the interoperability objective.

The first version of this use case is now under review. It describes the system development process plus the engineering methods that capture the detailed steps and the artifacts that are used and produced at each step. This gives an idea of the type of information that is to be considered for interoperability requirements.

Verum's role in the definition of this use case is, together with the other use case participants, to agree on the place of the ASD:Suite in the overall system engineering process, and to define the engineering methods and possible exchange of artifacts with other 'bricks' in the overall use case.

For the ASD:Suite, the interoperability requirements resulting from this use case are limited to requirements traceability. The exact details of these requirements will become clear in the next reporting period.

3.2.1.3 General improvement

The usability and 'reach' of the ASD:Suite will be improved by:

- Extending it with (scripting) interfaces; this enables the interoperability with other tools, for instance for linking model elements to external requirements or other system engineering artifacts.

- Extending it with feedback mechanisms of the runtime execution behavior, for instance to enable analysis of component interaction during runtime execution.
- Extending it with support for validation and simulation of components and (sub)systems; this enables users to check that the component/system indeed behaves as intended.
- Better visualization of the model and component overview to aid in architecture analysis and design.

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

3.2.1.4 Integration and interoperability

Based on the Volvo use case, first the use case is analyzed and the corresponding requirements are collected, harmonized and prioritized. Then these requirements are further developed and implemented into the ASD:Suite using an iterative approach. The resulting extensions of the ASD:Suite are then fed back into the Volvo use case for assessment and feedback. This will probably take a few cycles to arrive at an implementation that adds value to the use case and its participants.

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

3.2.2 B4.9 - Rapid design analysis (POOSL)

3.2.2.1 Description

Name:	Rapid design analysis – POOSL
Contact:	arjan.mooij@tno.nl
Dependencies	
License	
Additional information	

POOSL is an object-oriented language for specifying system behavior including parallelism. POOSL is used for rapid prototyping of functional system behavior as it enables the quick exploration of multiple design alternatives. In addition, it enables the analysis of performance characteristics of architectures. POOSL is currently supported by two tools:

- SHESim: model editor with a built-in interactive simulator;
- Rotalumis: high-performance simulation engine.

3.2.2.2 Use Case coverage and application

The requirements from the use cases in the health care domain are as follows:

- **Improve clarity of requirements**: currently, the requirements are described using informal text documents. The goal is to improve their clarity by making more precise and more visual descriptions of the requirements.
- **Early feedback on requirements, architecture, and design**: currently, the requirements, architecture, and design are described using documents. The goal is to get early feedback on these descriptions by making analyzable models that can be simulated.
- **Early integration testing**: currently, integration testing requires the availability of large amounts of software and hardware. The goal is to enable early integration testing by using executable models of the software and hardware that is not yet available.
- **Reduce testing on physical hardware**: currently, modifications of the software implementation should be tested on all product (hardware) configurations. The goal is to reduce the amount of testing on physical machines by testing the software in combination with hardware simulators.

3.2.2.3 General improvement

The usability of the POOSL tools will be improved by means of an Eclipse IDE. This includes early feedback to language users based on static model analysis, such as type checking. Moreover, it includes modularization techniques for managing the complexity of industrial scale models, and model conversions for interoperability with other POOSL tools. In particular, it includes convenient and interactive access to simulation results produced by the tool Rotalumis.

The tool Rotalumis will be extended with an external socket communication interface for interoperability between a simulated POOSL model and models in external simulation tools, such as physics models and visual simulations.

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

3.2.2.4 Integration and interoperability

POOSL will be used to create and simulate functional models of the software of an X-Ray scanner. Multiple models will be made to clarify and analyze the requirements, architecture and design documents. These models differ in the abstraction level, the parts of the system that are covered, and may take into account multiple design alternatives.

We will integrate these ingredients in order to obtain executable simulations of the X-Ray scanner in early development phases. A complete integration consists of the following parts that interoperate dynamically while performing the simulation:

- Functional model of the software (**POOSL**);
- Physics model of the hardware (e.g., Matlab, Modelica, Blender, etc.);
- Visual simulation of the hardware (**NobiVR**).

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

3.2.3 B4.1 - NobiVR

3.2.3.1 Description

Name:	NobiVR
Contact:	
Dependencies	
License	
Additional information	

NobiVR is a tool to visualize and simulate functional 3D models, implemented on top of a virtual reality (VR) layer. This VR layer can accommodate 3D motion tracking input for natural user interaction, and multiple types of immersive 3D VR configurations. NobiVR is used to visualize 3D data, both volume and geometry. Volume visualization is primarily applied to medical imaging data (CT/MRI/Ultrasound/etc), while geometry visualization is applied to many types of data such as medical segmentations, 3D scanner output, CAD files, etc. Any application which is based on the VR layer of NobiVR is configurable, allowing them to make use of many different VR hardware components ranging from desktop workstations to large projection setups by simply loading different configurations.

3.2.3.2 Use Case coverage and application

The requirements from the use cases in the health care domain are as follows:

- **Improve clarity of requirements**: currently, the requirements are described using informal text documents. The goal is to improve their clarity by making more precise and more visual descriptions of the requirements.
- **Early feedback on requirements, architecture, and design**: currently, the requirements, architecture, and design are described using documents. The goal is to get early feedback on these descriptions by making analyzable models that can be simulated.

- **Early integration testing**: currently, integration testing requires the availability of large amounts of software and hardware. The goal is to enable early integration testing by using executable models of the software and hardware that is not yet available.
- **Reduce testing on physical hardware**: currently, modifications of the software implementation should be tested on all product (hardware) configurations. The goal is to reduce the amount of testing on physical machines by testing the software in combination with hardware simulators.

3.2.3.3 General improvement

The visualization and VR functionality of **NobiVR** will be provided in this project. Improvements to these functionalities will be done in this project, as well as the implementation of a remote rendering engine in NobiVR

- to enable systems without powerful graphics hardware to use NobiVR, and
- to allow collaboration by multiple users at different locations on a shared instance of the VR layer.

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

3.2.3.4 Integration and interoperability

NobiVR will be used to create a visual simulation of the hardware of an X-Ray scanner. This environment will be used during the requirements, architecture and design phases. This simulation tool can produce visual output (images/videos) to support and/or replace the current textual recording of requirements.
We will integrate these ingredients in order to obtain executable simulations of the X-Ray scanner in early development phases. A complete integration consists of the following parts that interoperate dynamically while performing the simulation:

- Functional model of the software (**POOSL**);
- Physics model of the hardware (e.g., Matlab, Modelica, Blender, etc.);
- Visual simulation of the hardware (**NobiVR**).

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

3.2.4 Static code analysis

The term Static Code Analysis (SCA) has a broad meaning. It can encompass everything from syntax check over style compliance over check of programming conventions to logical or dynamical implications for the function. There are two bricks explicitly included in CRYSTAL for such purposes: Astrée and Polyspace. The two have a different profile due to historical reasons. AbsInt tools concentrate on offering dominating performance in analysis of logical and dynamical implications while Polyspace allocates some of its strength in compliance tests. Depending on the goals of the use-case one or the other should be integrated into the tool chain. However, it would be wrong to fully focus on these two tools alone.

3.2.4.1.1 Drawbacks of Conventional Tests

Businesses building mission/safety critical systems must assure themselves that certain qualities are good. The conventional practice is to perform a complete test-set for the implemented functions under a large variety of conditions. There are common characteristics to this approach:

- Tests become a significant financial burden.
- It is very difficult to test a specific conceptual middle ground because such tests become very complex - potentially more complex than the DUT (device under test) itself. Developing special purpose test suites can become a substantial task. Investments into this area must be protected through intensive reuse.
- Algorithmic spaces for complex systems are simply vast. It is very difficult to cover them sincerely.
- A broad standing test-set (which is quite an investment) is often detrimental to the decision to refactor you code in order to improve its architecture ("adverse effect of testing"). Something more flexible is needed.

The problems experienced with testing have led to the idea that it would be more efficient to simply prove the correctness of a given solution. But the property "correctness" is not an attribute of the code itself but rather an attribute of the relationship between code and expectations. "Correctness" simply means a match between expectations, assumptions and solution descriptions. When applying proving techniques a general awareness of this fact should be maintained.

3.2.4.1.2 Alternatives to Testing

Static code analysis does not execute the program but looks into the source code for consequences. The analysis is performed automatically by tools like Lint, Astree or Polyspace. SCA is the successor of the classical code review which can be quite laborious and often impractical for large systems. The source code is checked by a series of formal techniques, e.g. abstract interpretation, in order to find several types of errors.

In contrast to other test methods, static verification covers the complete use of value ranges of input signals, parameters and maps. This can rule out run-time errors for various conditions. A limit analysis is also automatically covered. However, static verification cannot replace functional testing but it can eliminate a large body of tests designed to detect implementation weaknesses.

According to **our experience** a good mixture of static **tests** and functional tests will yield an excellent **price** per feature and per achieved **reliability**. Introducing provers to your testing landscape makes the handling of the processes even more difficult. If they can be made interoperable with other tools based on a common technology like OSLC then it would become more attractive to use them.

3.2.5 B3.51 - AbsInt

3.2.5.1 Description

Name:	AbsInt
Contact:	Aleksander.Lodwich@itk-engineering.de

Dependencies	
License	
Additional information	

AbsInt is offering a series of cutting-edge products for C-Code analysis. AbsInt analyzers are not model-based but are sometimes used in tool-chains with models at their top. They help to verify the absence of run-time errors and they help to estimate the amount of required resources on embedded systems.

AbsInt has become well known for his static code analyzer Astrée. The tool was designed with a zero false warnings mindset. It is capable of providing hints to the analyzer from source code. The software engineer can communicate detailed assumptions about his code without having programmed them out explicitly. This helps Astrée to interpret the code appropriately without exaggerated caution. From our experience Astrée is the fastest high quality analyzer to date and orders of magnitude faster than the immediate competitor Polyspace.

AbsInt also offers tools for investigating timing relationships between software and hardware on real-time systems (WCET analysis). Such tools are very helpful when trying to optimize systems for energy consumption or for evaluating the effect of two different ECU platforms. For example the energy consumption of an ECU could be reduced by lowering clock frequency but undesired deadlocks or errors could result from bad timing. AbsInt analyzers help to estimate the lower bounds for clock speeds in such case. Such estimates derived from code structure are far more solid than extensive tests and cheaper to do so as well.

3.2.5.2 Use Case coverage and application

We provide service to all use-cases and consulting to all work-packages in SP6 regarding the application and interoperability improvements of AbsInt tools. In the AbsInt brick ITK Engineering is predominantly offering experiences made with Astrée. At the moment no demand has been formulated from use-cases. Use-Case 3.2 has declared interest into static code analysis.

3.2.5.3 General improvement

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

3.2.5.4 Integration and interoperability

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD

Description: TBD	
Link to internal working documents:	

3.2.6 B3.47 - Mathworks Polyspace

3.2.6.1 Description

Name:	Mathworks Polyspace (B3.47)
Contact:	Aleksander.Lodwich@itk-engineering.de
Dependencies	
License	
Additional information	

Mathworks' Polyspace is a popular package for static code analysis of C/C++ and Ada-Code. Polyspace can check code for its potential to produce run-time errors, overflows and the like. The tool offers reporting functionality and it is frequently found in automated build processes in industry. Polyspace can compare code with predefined coding rules and helps to track development progress. The tool can be extended with a certification extension. With this extension it is possible to certify certain code revisions for meeting ISO 26262, IEC 61508 or DO-178B standards. This makes it a very popular tool in respective domains.

Just like AbsInt analyzers, Polyspace is not a tool to be qualified as "model-based". However, it is often found in tool chains with models as the source. It will add trust to your project when a new compiler is used, when refactoring was done or when generation parameters were changed.

However, Polyspace is defensive and will assume the worst during analysis. The consequence of this approach is a large number of false warnings which have to be checked and tracked manually. As the number of warnings can become very large, many tool-chain authorities shy its inclusion because of the missing resources to handle the result validation. The reports have to be managed and effective report life-cycle maintained. This proves to be not so convenient in real life but it could be greatly simplified by making Polyspace interoperate with issue trackers. Unfortunately, since the warning and error generation details are closed source some uncertainty remains in this approach. We would use our role as distinguished partner of Mathworks to improve the situation here. Polyspace could for example generate id-codes from analyzed structure which would make the process of tracking report entities easier.

3.2.6.2 Use Case coverage and application

We provide service to all use-cases and consulting to all work-packages in SP6 regarding the application and interoperability improvements of Polyspace. ITK offers extensive practical experience with Polyspace. At the moment no demand has been formulated from use-cases. Use-Case 3.2 has declared interest into static code analysis.

3.2.6.3 General improvement

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

3.2.6.4 Integration and interoperability

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

3.2.7 B4.15 - Interoperable architectural analysis

3.2.7.1 Description

Name:	Interoperable architecture analysis
Contact:	elluna@iti.es, rjuan@iti.es
Dependencies	
License	
Additional information	

Interoperable architectural analysis tool provide means to perform ICT interoperable architectural requirements analysis at early stages through the modeling of the product. Thus, this tool allows evaluating the ICT interoperability of the components of the product checking the existence of conflicting interfaces at hardware and software level. In this way the tool provides means for guaranteeing the correct integration of components (or new components) in (within) the product at early stages in the product development process.

3.2.7.2 Use Case coverage and application

The Interoperable Architectural Analysis tool will be used in UC406 for determining at early stages that interoperability requirements between the infusion pump (provided by a supplier) and the other components of the system can work together.

Use Case 4.06 is titled: “An intelligent infusion controller for Blood Pressure regulation in Operating Room (OR)”, and its goal is to incorporate tools to support certain phases of the development of an intelligent infusion controller.

The product to be obtained in this process is a system that operates delivering vasoactive drugs with the ultimate goal of reducing patient´s hypertension, and precisely controlling blood pressure measurements in a patient undergoing surgical intervention in Operating Room or in postcardiac surgery in ICU (intensive care unit).

The use case owner, RGB medical provides the following basic requirements:

- The tool shall detect ICT interoperable issues between different components of the product, especially those existing issues between components developed internally and components provided by suppliers. This detection must be done in the early stages of the development process for minimizing costs derived from a late detection.
- The tool shall provide a graphic modeler that supports rapid and assisted modeling of the components.
- The tool shall support the mapping of modeled components with the product requirements. In this way the solution supports the identification of non-fulfilled requirements based on the ICT interoperability issues detected.
- The tool shall generate reports with the obtained evaluation results. This report must identify the detected ICT interoperability issues and possible potential issues.
- The tool shall be based on well-established languages and technologies in order to avoid dependencies on obsolete and/or abandoned solutions that jeopardize its use and maintenance.
- The tool shall be easy maintainable and provide a long-time period of maintenance.
- The tool shall be able to run at least on Windows systems.

The preliminary IOS requirements from this use case point of view are related to improve the requirements traceability during the project. The following list details the currently identified requirements:

- The basic IOS requirement consists in being able to import requirements from a requirement management tool. This tool must be able to connect to a requirement management tool for importing the current list of requirements. Later the tool should provide functionality for linking the imported requirement with the components in the model in charge of satisfying them. This mapping must to perform a follow up of the requirements fulfillment from an ICT interoperability point of view.
- As advanced IOS requirement, the tool shall be able to generate status information about the fulfillment of each ICT interoperable requirement and functionality for including this information in the Requirements Management tool.

3.2.7.3 General improvement

During the CRYSTAL project it will be implemented and provided the basic tool functionality and the IOS basic required support.

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

3.2.7.4 Integration and interoperability

This brick will be used in the “Rapid prototyping of architecture and design” phase of the V process model of their use case. This phase is done once a first Requirement Specification has been achieved, and before starting the “Modular Decomposition” phase.

Regarding tools integration, from IOS Requirements it is derived that this brick must be able to interact through OSLC with a Requirements Management tool. Therefore, support for OSLC Core and Requirements Management specification is needed.

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

3.2.8 B2.55 - Scheduling requirement analysis

3.2.8.1 Description

Name:	Scheduling requirement analysis
Contact:	iripoll@iti.es, rjuan@iti.es
Dependencies	
License	
Additional information	

This brick will provide means to analyze timing requirements for complex systems. In order to achieve this it will include a parser to manipulate AADL models which has been released by SAE as the aerospace standard AS5506. It will also facilitate an editor and analysis feature that performs the scheduling analysis of a partitioned system allowing working with incomplete models and allowing modification of the system incrementally. Thus this tool provides means for performing simulation analysis at the early stages of the product development in order to check the validity of the proposed solution from the scheduling requirements perspective.

3.2.8.2 Use Case coverage and application

This tool will be used in the UC205 CRYSTAL Space Toolset applied to Avionics Control Unit Software generation, test, V&V, and Certification.

This use case has as main goal is the implementation of the software for an Avionics Control Unit including autonomous navigation features based on GPS, inertial and/or image acquisition inputs. This unit will be based in a LEON architecture running in multicore configuration inside an FPGA.

In this development process usually are involved several actors. On one hand there is the hardware manufacturer, which purchases the software embedded in his units from external suppliers and then integrates it in the hardware. In this case, the hardware manufacturer remains responsible in front of the

customer of the quality and performances of the software embedded in the units. In addition to this all the code generated has to undergo an Independent Software Verification and Validation process (ISVV) by a third party company, typically selected and/or approved by the final customer.

The basic requirements provided by the use case are:

- The brick shall also include an editor and analysis feature that performs the scheduling analysis of a partitioned system allowing working with incomplete models and allowing modification of the system incrementally.
- The tool shall generate: a) Textual and graphical information system and generated plan, b) Generate the XML configuration file (ARINC-653) containing the partition c) Build plans for the tasks of each partition or scheduling priorities, and d) WCET and CPU budget.
- The tool shall be based on well-established languages and technologies in order to avoid dependencies on obsolete and/or abandoned solutions that jeopardize its use and maintenance.
- The tool shall accept heterogeneous HW systems (including memory architecture).
- The tool shall be independent of the RTOS and/or the underlying application executive (if any).
- The tool should provide ARINC-653 services configuration.
- The tool should be easy to maintain and provide a long-time period of maintenance.
- The tool shall be able to run at least on Windows systems.

The preliminary IOS requirement coming from this UC is that the tool shall communicate to a test tool the ARINC-653 services configuration to be tested (or another FDIR / safety tests schema TBD).

3.2.8.3 General improvement

During the CRYSTAL project it will be implemented and provided the basic tool functionality and the IOS basic required support.

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

3.2.8.4 Integration and interoperability

This brick will be used in the design phase of the V process model of this use case. This phase is done once a first Requirement Specification has been achieved, and before starting the “Modular Decomposition” phase.

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

4 Terms, Abbreviations and Definitions

CRYSTAL	**CR**itical S**YST**em Enginieering **A**cce**L**eration
R	Report
P	Prototype
D	Demonstrator
O	Other
PU	Public
PP	Restricted to other program participants (including the JU).
RE	Restricted to a group specified by the consortium (including the JU).
CO	Confidential, only for members of the consortium (including the JU).
WP	Work Package
SP	Subproject
UC	Use Case
SoP	State of Practice
SoA	State of the Art
MBT	Model-Based Testing
SME	Small and Medium-sized Enterprises
SUT	System under Test
MDT	Model-Driven Testing
MT	Model Transformations
DSML	Domain-Specific Modeling Language
DSM	Domain-Specific Modeling
UML	Unified Modeling Language
M2M	Model-to-Model
M2T	Model-to-Text
AMMA	ATLAS Model Management Architecture
MDA	Model Driven Architecture
MDE	Model-Driven Engineering
PIM	Platform Independent Model
PSM	Platform Specific Model
PST	Platform Specific Test
TTCN-3	Testing and Test Control Notation version 3
ETCS	European Train Control System
CNL	Controlled Natural Language
WLTP	Worldwide harmonized Light vehicles Test Procedures
ICU	Intensive Care Unit

Table 4-1: Terms, Abbreviations, and Definitions

5 References

[Utting, 2007]	M. Utting, B. Legeard; *Practical Model-Based Testing - a Tools Approach*; Morgan Kaufmann Publishers Inc. San Francisco, (2007)
[1020 WG, 1987]	ANSI/IEEE Std 1012-1986; *IEEE Standard for Software Verification and Validation Plans*; The Institute of Electrical and Electronics Engineers, Inc., February 10, 1987.
[Dai, 2004]	Z. Dai. *Model-driven testing with UML 2.0*; In Proceedings of the 2nd European Workshop on Model Driven Architecture, 2004.
[Utting, 2012]	M. Utting, A. Pretschner, B. Legeard; *A taxonomy of model-based testing*; STVR 22:5, 2012.
[Zander, 2011]	J. Zander, I. Schieferdecker, and P. J. Mosterman; *A Taxonomy of Model-Based Testing for Embedded Systems;* CRC Press, Boca Raton (2011)
[Mussa, 2009]	M. Mussa, S. Ouchani, W. Al Sammane, A.Hamou-Lhadj; *A Survey of Model-Driven Testing Techniques*; In IEEE Procs. of the 9th International Conference on Quality Software (QSIC '09), pp.167-172, 2009.
[Nguyen, 2003]	H. Q. Nguyen, et al.; *Testing Applications on the Web: Test Planning for Internet-Based Systems*; John Wiley & Sons. 2003
[Abran, 2004]	A. Abran et al., eds. *Guide to the Software Engineering Body of Knowledge*; IEEE Computer Society, 2004
[Linzhang, 2004]	W. Linzhang, Y. Jiesong, Y. Xiaofeng, H. Jun, L. Xuandong, Z. Guoliang; *Generating test cases from UML activity diagram based on Gray-box method*; Software Engineering Conference, 2004. 11th Asia-Pacific , pp.284,291, 30 Nov.-3 Dec. 2004 doi: 10.1109/APSEC.2004.55
[Baharom, 2008]	S. Baharom, Z. Shukur; *Module documentation based testing using Grey-Box approach*; Information Technology, 2008. ITSim 2008. International Symposium on , vol.2, pp.1,6, 26-28 Aug. 2008doi: 10.1109/ITSIM.2008.4631651
[Dadeau, 2011]	F. Dadeau, F. Peureux; *Grey-Box Testing and Verification of Java/JML*; Software Testing, Verification and Validation Workshops (ICSTW), 2011 IEEE Fourth International Conference on , pp.298,303, 21-25 March 2011 doi: 10.1109/ICSTW.2011.30
[Petrenko, 1995]	A. Petrenko, N. Yevtushenko, and R. Dssouli; *Testing Strategies for Communicating FSMs in Protocol Test Systems*; IFIP — The International Federation for Information Processing, 1995, doi=10.1007/978-0-387-34883-4_13
[Dong, 2009]	Y.-W. Dong; G. Wang; H.-B. Zhao; *A Model-Based Testing for AADL Model of Embedded Software*; Quality Software, 2009. QSIC '09. 9th International Conference on , pp.185,190, 24-25 Aug. 2009 doi: 10.1109/QSIC.2009.33
[CENELEC, 2004]	CENELEC, *EN50128 - Railway applications, communications, signalling and processing systems - Software for railway control and protection systems*, 2004
[IEC, 1998]	IEC, *IEC Publication 61508-3, Functional safety of electrical/electronic/programmable electronic safety-related systems*, Part 3: Software requirements, 1998
[De Nicola, 2005]	G. De Nicola, P. di Tommaso, R. Esposito, F. Flammini, P. Marmo, A. Orazzo; *A Grey-Box Approach to the Functional Testing of Complex Automatic Train Protection Systems*; Dependable Computing - EDCC 5, Lecture Notes in Computer Science Volume 3463, 2005, pp 305-317.
[Piper, 2012]	T. Piper, S. Winter, P. Manns, N. Suri; *Instrumenting AUTOSAR for dependability assessment: A guidance framework*; Dependable Systems and Networks (DSN), 2012 42nd Annual IEEE/IFIP International Conference on , vol., no., pp.1,12, 25-28 June 2012doi: 10.1109/DSN.2012.6263913
[OMG, 2003]	Object Management Group; *Model driven architecture*; V. 1.0.1, 2003
[Mellor, 2003]	S.J. Mellor, A.N. Clark, T. Futagami; *Model-driven development - Guest editor's introduction*; Software, IEEE , vol.20, no.5, pp.14,18, Sept.-Oct. 2003 doi: 10.1109/MS.2003.1231145

[Schmidt, 2006]	D.C. Schmidt; *Model-driven engineering*; IEEE COMPUTER, 39(2):25, 2006.
[Kelly, 2008]	S. Kelly and J.-P. Tolvanen; *Domain-specific modeling: enabling full code generation*; Wiley-IEEE Computer Society Press, 2008.
[OMG, 2011]	Object Management Group; *UML2: Unified Modeling Language: Infrastructure and Superstructure*; May 2011. Version 2.4, formal/11-08-05.
[Fuentes, 2004]	L. Fuentes-Fernández and A. Vallecillo-Moreno; *An introduction to UML profiles; UML and Model Engineering*, 2, 2004.
[OMG, 2012]	Object Management Group; *OCL2: Object Constraint Language*; January 2012. Version 2.3, formal/12-01-01.
[Giachetti, 2009]	G. Giachetti, B. Marin, and O. Pastor; *Integration of domain-specific modeling languages and UML through UML profile extension mechanism*; International Journal of Computer Science and Applications, 6(5):145–174, 2009.
[Selic, 2007]	B. Selic; *A systematic approach to domain-specific language design using UML*; In Object and Component-Oriented Real-Time Distributed Computing, 2007. ISORC'07. 10th IEEE International Symposium on, pages 2–9. IEEE, 2007.
[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
[Myers, 2011]	G.J. Myers, C. Sandler, T. Badgett ; *The Art of Software Testing; 3rd Edition*, Wiley , ISBN: 978-1-118-03196-4, December 2011
[Dalal, 1999]	S.R. Dalal, A. Jain, N. Karunanithi, J.M. Leaton, C.M. Lott, G.C. Patton; *Model-Based Testing in Practice*; Proceeding of ICSE'99 (ACM Press) (Proceedings of ICSE'99 (ACM Press)), 1999.
[Tretmans, 2003]	J. Tretmans, E. Brinksma; *TorX : Automated Model Based Testing*; First European Conference on Model-Driven Software Engineering, 2003: 31-43.
[Gargantini, 1999]	A. Gargantini, C. Heitmeyer; *Using model checking to generate tests from requirements specifications*; ESEC/FSE-7 Proceedings of the 7th European software engineering conference held jointly with the 7th ACM SIGSOFT international symposium on Foundations of software engineering, 1999: 146-162 .
[Javed, 2007]	A.Z. Javed, P.A. Strooper, G.N. Watson; *Automated generation of test cases using model driven architecture*; In Proc. of the ICSE 2nd International Workshop on Automation of Software Test (AST), 2007.
[Mingsong, 2006]	Mingsong C., Xiaokang Q., Xuandong L.; *Automatic Test Case Generation from UML Activity*; In Proc. of the International Workshop on Automation of software test, 2006: 2-8.
[Crichton, 2007]	C. Crichton, A. Cavarra, J. Davies;*Using UML for Automatic Test Generation*; In Proc. of the Automation of Software Testing, 2007.
[McMinn, 2004]	P. McMinn; *Search-based software test data generation : a survey*; Software testing, verification and reliability (STVR), 2004: 105-156.
[Tonella, 2004]	P., Tonella; *Evolutionary testing of classess*; ISSTA '04 : proceeding of the 2004 ACM SIGSOFT international symposium on Software Testing and Analysis, 2004: 119.128.
[Fischer, 1982]	K.F., Fischer; *A test case selection method for the validation of software maintenance modification*; Proceedings COMPSAC82, 1982: 529-537.
[Mussa, 2009]	M. Mussa, S. Ouchani, W. Al Sammane, A. Hamou-Lhadj; *A Survey of Model-Driven Testing Techniques*; Quality Software, 2009. QSIC '09. 9th International Conference on
[Sendall 2003]	Sendall, S. ; Kozaczynski, W. *Model Transformation: the Heart and Soul of Model-Driven Software Development* Software, IEEE (Volume:20 , Issue: 5), Sept.-Oct. 2003
[Tómasson 2013]	H. Tómasson, H. Neukirchen; *Distributed testing of cloud computing applications using the TTCN-3-based Jata test framework*; Proceeding NordiCloud '13 Proceedings of the Second Nordic Symposium on Cloud Computing & Internet echnologies Pages 22-29
[Hecker 2003]	R. Heckel, M. Lohmann; *Towards Model-Driven Testing*; Electronic Notes in Theoretical Computer Science Volume 82, Issue 6, September 2003, Pages 33–43
[Guelfi, 2008]	N. Guelfi, B. Ries; *SESAME: A Model-Driven Test Selection Process for Safety-Critical*

	Embedded Systems; ERCIM News 2008
[Bouquet 2007]	F. Bouquet, C. Grandpierre, B. Legeard, F. Peureux, N. Vacelet, and M. Utting; *A subset of precise UML for Model-based Testing*; In Proc. of the 3rd International Workshop Advances in Model Based Testing (AMOST), pp. 95-104, 2007.
[Myers, 1979]	G. J. Myers; The Art of Software Testing; Wiley, New York (1979)
[Utting, 2007]	M. Utting, B. Legeard; Practical Model-Based Testing - a Tools Approach; Morgan Kaufmann Publishers Inc. San Francisco, (2007)
[IEEE, 1987]	ANSI/IEEE Std 1012-1986; IEEE Standard for Software Verification and Validation Plans; The Institute of Electrical and Electronics Engineers, Inc., February 10, 1987.
[DAI, 2004]	Z. Dai. Model-driven testing with UML 2.0. In Proceedings of the 2nd European Workshop on Model Driven Architecture, 2004.
[Utting, 2012]	M. Utting, A. Pretschner, B. Legeard; A taxonomy of model-based testing; STVR 22:5, 2012.
[Zander, 2011]	J. Zander, I. Schieferdecker, and P. J. Mosterman;A Taxonomy of Model-Based Testing for Embedded Systems (2011)
[Mussa, 2009]	M. Mussa, S. Ouchani, W. Al Sammane, A.Hamou-Lhadj; A Survey of Model-Driven Testing Techniques, In IEEE Procs. of the 9th International Conference on Quality Software (QSIC '09),pp.167-172, 2009.
[Nguyen, 2003]	Nguyen, Hung Q, et al. Testing Applications on the Web: Test Planning for Internet-Based Systems. John Wiley & Sons. 2003
[Abran, 2004]	A.Abran et al., eds. Guide to the Software Engineering Body of Knowledge, IEEE Computer Society, 2004
[Linzhang, 2004]	Wang Linzhang; Yuan Jiesong; Yu Xiaofeng; Hu Jun; Li Xuandong; ZhengGuoliang, "Generating test cases from UML activity diagram based on Gray-box method," Software Engineering Conference, 2004. 11th Asia-Pacific , vol., no., pp.284,291, 30 Nov.-3 Dec. 2004doi: 10.1109/APSEC.2004.55
[Baharom, 2008]	Baharom, S.; Shukur, Z., "Module documentation based testing using Grey-Box approach," Information Technology, 2008. ITSim 2008. International Symposium on , vol.2, no., pp.1,6, 26-28 Aug. 2008doi: 10.1109/ITSIM.2008.4631651
[Dadeau, 2011]	Dadeau, F.; Peureux, F., "Grey-Box Testing and Verification of Java/JML," Software Testing, Verification and Validation Workshops (ICSTW), 2011 IEEE Fourth International Conference on , vol., no., pp.298,303, 21-25 March 2011doi: 10.1109/ICSTW.2011.30
[Petrenko, 1995]	Petrenko, A. and Yevtushenko, N. and Dssouli, R., Testing Strategies for Communicating FSMs in Protocol Test Systems,IFIP — The International Federation for Information Processing, 1995, doi=10.1007/978-0-387-34883-4_13
[Dong, 2009]	Yun-wei Dong; Geng Wang; Hong-bing Zhao, "A Model-Based Testing for AADL Model of Embedded Software," Quality Software, 2009. QSIC '09. 9th International Conference on , vol., no., pp.185,190, 24-25 Aug. 2009doi: 10.1109/QSIC.2009.33
[CENELEC, 2004]	CENELEC, EN50128 - Railway applicationsCommunications, signalling and processing systems - Software forrailway control and protection systems, 2004
[IEC, 1998s]	IEC, IEC Publication 61508-3, Functional safety ofelectrical/electronic/programmableelectronic safety-related systems, Part 3: Software requirements, 1998
[De Nicola, 2005]	Giuseppe De Nicola, Pasquale di Tommaso, Esposito Rosaria, Flammini Francesco, MarmoPietro, Orazzo Antonio, A Grey-Box Approach to the Functional Testing of Complex Automatic Train Protection Systems, Dependable Computing - EDCC 5, Lecture Notes in Computer Science Volume 3463, 2005, pp 305-317
[Piper, 2012]	Piper, T.; Winter, S.; Manns, P.; Suri, N., "Instrumenting AUTOSAR for dependability assessment: A guidance framework," Dependable Systems and Networks (DSN), 2012 42nd Annual IEEE/IFIP International Conference on , vol., no., pp.1,12, 25-28 June 2012doi: 10.1109/DSN.2012.6263913
[S1]	S. Arora, A. Gadkari, and S. Ramesh. Scenario-Based Specification of Automotive Requirements With Quantitative Constraints and Synthesis of SL/SF Monitors.

	Embedded Systems Letters, IEEE, 3(2):62–65, 2011.
[S2]	T. Bauer, T. Beletski, F. Bohr, R. Eschbach, D. Landmann, and J. Poore. From Requirements to Statistical Testing of Embedded Systems. In *Proceedings of Fourth International Workshop on Software Engineering for Automotive Sys- tems, 2007*, pages 3–3. IEEE, 2007.
[S3]	J.-L. Boulanger and V. Q. Dao. Requirements engineering in a model-based methodology for embedded automotive software. In *Proceedings of IEEE Inter- national Conference on Research, Innovation and Vision for the Future, 2008*, pages 263–268. IEEE, 2008.
[S4]	F. Bouquet, E. Jaffuel, B. Legeard, F. Peureux, and M. Utting. Requirements traceability in automated test generation. *ACM SIGSOFT Software Engineering Notes*, 30(4):1, July 2005.
[S5]	M. Cecconi and E. Tronci. Requirements formalization and validation for a telecommunication equipment protection switcher. In *Proceedings of HASE 2000 High Assurance Systems Engineering, 2000, Fifth IEEE International Symposim on*, pages 169–176. IEEE, 2000.
[S6]	A. Cimatti, M. Roveri, A. Susi, and S. Tonetta. Validation of requirements for hybrid systems. *ACM Transactions on Software Engineering and Methodology*, 21(4):1–34, Nov. 2012.
[S7]	S. Easterbrook, R. Lutz, R. Covington, J. Kelly, Y. Ampo, and D. Hamilton. Experiences using lightweight formal methods for requirements modeling. *Software Engineering, IEEE Transactions on*, 24(1):4–14, 1998.
[S8]	A. El-Ansary. Requirements Definition of Safe Software Using the Behavioral Patterns Analysis (PBA) Approach: The Railroad Crossing System. In *Com- putational Intelligence for Modelling, Control and Automation, 2006 and In- ternational Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, pages 80–80. IEEE, 2006.
[S9]	W. Fleisch. Applying use cases for the requirements validation of component-based real-time software. In *Proceedings of 2nd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, 1999*, pages 75–84. IEEE, 1999.
[S10]	B. Fontan, L. Apvrille, P. De Saqui-Sannes, and J.-P. Courtiat. Real-Time and Embedded System Verification Based on Formal Requirements. In *Proceedings of International Symposium on Industrial Embedded Systems, 2006*, pages 1–10. IEEE, 2006.
[S11]	R. Grosu, I. Kruger, and T. Stauner. Requirements specification of an automotive system with hybrid sequence charts. In *Proceedings of Fifth International Workshop on Object-Oriented Real-Time Dependable Systems, 1999*, pages 149–151. IEEE, 1999.
[S12]	M. P. E. Heimdahl and N. G. Leveson. Completeness and consistency analysis of state-based requirements. In *Proceedings of the 17th international conference on Software engineering - ICSE '95*, pages 3–14, New York, New York, USA, Apr. 1995. ACM Press.
[S13]	C. Heitmeyer, J. Kirby, and B. Labaw. The SCR method for formally specifying, verifying, and validating requirements. In *Proceedings of the 19th international conference on Software engineering - ICSE '97*, pages 610–611, New York, New York, USA, May 1997. ACM Press.
[S14]	X. Jin, A. Donz´e, J. V. Deshmukh, and S. A. Seshia. Mining requirements from closed-loop control models. In *Proceedings of the 16th international conference on Hybrid*

	systems: computation and control - HSCC '13, page 43, New York, New York, USA, Apr. 2013. ACM Press.
[S15]	C. Knieke, M. Huhn, and M. Lochau. Modeling and Validation of Executable Requirements Using Live Activity Diagrams. In *Proceedings of Sixth International Conference on Software Engineering Research, Management and Applications, 2008*, pages 51–58. IEEE, 2008.
[S16]	S. Konrad and B. H. C. Cheng. Requirements patterns for embedded systems. In *Proceedings of IEEE Joint International Conference on Requirements Engi- neering, 2002*, pages 127–136. IEEE, 2002.
[S17]	J.-S. Lee and S.-D. Cha. Behavior verification of hybrid real-time requirements by qualitative formalism. In *Proceedings., Fourth International Workshop on Real-Time Computing Systems and Applications, 1997*, pages 127–134. IEEE, 1997.
[S18]	K.-H. Lee, P.-G. Min, J.-H. Cho, and D.-J. Lim. Model-driven requirements validation for automotive embedded software using UML. In *Computing Technology and Information Management (ICCM), 2012 8th International Conference on*, pages 46–50. IEEE, 2012.
[S19]	R. Löffler, M. Meyer, and M. Gottschalk. Formal scenario-based requirements specification and test case generation in healthcare applications. In *Proceedings of the 2010 ICSE Workshop on Software Engineering in Health Care - SEHC '10*, pages 57–67, New York, New York, USA, May 2010. ACM Press.
[S20]	F. Mallet, M.-A. Peraldi-Frati, and C. Andre. Marte CCSL to Execute East-ADL Timing Requirements. In *Proceedings of IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, 2009*, pages 249–253. IEEE, 2009.
[S21]	F. Merz, C. Sinz, H. Post, T. Gorges, and T. Kropf. Abstract Testing: Connecting Source Code Verification with Requirements. In *Quality of Information and Communications Technology (QUATIC), 2010 Seventh International Conference on the*, pages 89–96. IEEE, 2010.
[S22]	E. Nasr, J. McDermid, and G. Bernat. Eliciting and specifying requirements with use cases for embedded systems. In *Proceedings of the Seventh International Workshop on Object-Oriented Real-Time Dependable Systems, 2002*, pages 350–357. IEEE, 2002.
[S23]	A. Post, J. Hoenicke, and A. Podelski. Vacuous real-time requirements. In *Requirements Engineering Conference (RE), 2011 19th IEEE International*, pages 153–162. IEEE, 2011.
[S24]	H. Post, C. Sinz, F. Merz, T. Gorges, and T. Kropf. Linking Functional Requirements and Software Verification. In *Proceedings of 17th IEEE International Requirements Engineering Conference, 2009*, pages 295–302. IEEE, 2009.
[S25]	F. Schneider, S. Easterbrook, J. Callahan, and G. Holzmann. Validating requirements for fault tolerant systems using model checking. In *Proceedings of 1998 Third International Conference on Requirements Engineering, 1998*, pages 4–13. IEEE, 1998.
[S26]	P. Shaker. Feature-oriented requirements modelling. In *Software Engineering, 2010 ACM/IEEE 32nd International Conference on*, pages 365–368. IEEE, 2010. [S27] S. Siegl, K.-S. Hielscher, and R. German. Model Based Requirements Analysis and Testing of Automotive Systems with Timed Usage Models. In *Requirements Engineering Conference (RE), 2010 18th IEEE International*, pages 345–350. IEEE, 2010.
[S27]	S. Siegl, K.-S. Hielscher, and R. German. Model Based Requirements Analysis and Testing of Automotive Systems with Timed Usage Models. In Requirements Engineering Conference (RE), 2010 18th IEEE International, pages 345-350. IEEE, 2010.

[S28]	S. Siegl, K.-S. Hielscher, R. German, and C. Berger. Automated testing of embedded automotive systems from requirement specification models. In *Test Workshop (LATW), 2011 12th Latin American*, pages 1–6. IEEE, 2011.
[S29]	M. Winokur, J. Lavi, I. Lavi, and R. Oz. Requirements analysis and specification of embedded systems using ESCAM-a case study. In *Proceedings of the 1990 IEEE International Conference on Computer Systems and Software Engineering CompEuro '90*, pages 80–89. IEEE, 1990.
[S30]	S. Zafar and R. Dromey. Integrating safety and security requirements into design of an embedded system. In *Proceedings of 12th Asia-Pacific Software Engineer- ing Conference, 2005*, page 8 pp. IEEE, 2005.
[Daramola, 2012]	Daramola O., Sindre G., Stalhane T.; Pattern-Based Security Requirements Specification Using Ontologies and Boilerplates, IEEE International Workshop on Requirements Patterns, pp. 54-59, 2012
[Denger, 2003]	Denger C., Berry D., Kamsties E.; Higher Quality Requirements Specifications through Natural Language Patterns, In: Proc. of IEEE Conference on Software-Science, Technology & Engineering, pp. 80-91, 2003
[Farfeleder, 2011a]	Farfeleder S., Moser T., Krall A., Stahlhane T., Zojer H., Panis C.; DODT: Increasing requirements formalism using domain ontologies for improved embedded systems development; In 14th IEEE Symposium on DDECS, pp. 1-4, 2011
[Farfeleder, 2011b]	Farfeleder S., Moser T., Krall A., Stahlhane T., Omoronyia I., Zojer H.; Ontology-Driven Guidance for Requirements Elicitation; In Proc. 8th ESWC, vol. 6644, pp. 212.226, 2011
[Hull, 2011]	Hull E., Jackson K., Dick J.; Requirements Engineering; Springer, 2011
[Holtmann, 2010]	Holtmann J.; Mit Satzmustern von textuellen Anforderungen zu Modellen; OBJEKT-spektrum RE/2010, 2010
[Holtmann, 2011]	Holtmann J., Meyer J., von Detten M.; Automatic Validation and Correction of Formalized, Textual Requirements, In: Proc. of 4th IEEE Conference on Software Testing, pp. 486-495, 2011
[Reinkemeier, 2011]	Reinkemeier P., Stierand I., Rehkop P., Henkler S.; A pattern-based requirement specification language: Mapping automotive specific timing requirements; In: Proc. of Software Engineering 2011, Lecture Notes in Informatics, pp.99-108; 2011
[Videira, 2005]	Videira C., Rodrigues Da Silva A.; Patterns and metamodel for a natural-language-based requirements specification language, In Proc. CaiSE'05, pp. 189-194, 2005
[Yue, 2011]	Yue T., Briand L. C., Labiche Y.; A systematic review of transformation approaches between user requirements and analysis models, In Requirements Engineering 16, pp. 75-99, Springer, 2011

6 Annex

6.1 Annex I: Survey questions

0% completed

CRYSTAL

Rationale:
The aim of the questionnaire is to get an overview of the state of practice and the needs regarding model-based engineering and model-based system analysis.
This means that we want to know if model-based engineering is used in practice, how it is used in development, and how models are used for a first verification/validation of the system.

Expected result:
The evaluation of the questionnaire should result in the derivation of open issues which should be covered within the CRYSTAL project funded by the EU.

Target group:
We want to target software architects, developers, project managers, system engineers, etc. from OEMs and suppliers from the embedded systems domain. This can, but not necessarily has to, include safety-critical systems.

Additional information:
The survey consists of 24 questions and will take approximately 15 minutes to answer. Additional 'other' fields can be added dynamically to some questions by filling in answers into the existing 'other' fields.
In some questions we ask about 'your division/department'. With the term division/department, we refer to organisational units with up to 50 people.

Your answers are treated completely anonymous.

The survey ends on 1st December 2013.

Next

Contact persons:
Grischa Liebel (Chalmers University of Technology), grischa@chalmers.se
Daniel Sauter (Itk Engineering) Daniel.Sauter@itk-engineering.de
Jos Langen (Verum), jos.langen@verum.com
Nadja Marko (VIRTUAL VEHICLE Research Center), Nadja.Marko@v2c2.at

20% completed

Context (Domain/Products/Terms)

1. In which company do you work?

2. Is your company a Small or Medium Enterprise (SME, <= 250 employees) or a large company (or part of)?

- ○ SME
- ○ Large Company
- ○ University/Research Institute
- ○ Other:

3. In which domain do you work?

- ☐ Automotive
- ☐ Avionics
- ☐ Healthcare
- ☐ Rail
- ☐ Domain independent
- ☐ Defense
- ☐ Telecommunications

Other:

4. What is your company's position in the value chain?

- ☐ OEM (Original Equipment Manufacturer – refers to a company that makes a final product for the consumer marketplace)
- ☐ First tier supplier (a company that is a direct supplier to OEMs)
- ☐ Second tier supplier (a company that is the key supplier to tier one suppliers)

Other:

5. How large is a work group within a typical project in your company?

- ○ 0-5 persons
- ○ 5-15 persons

- [ ] >15 persons

6. What are your main working tasks?

Architecture definition: ***We define architecture as the coarse grained structure of the system and how the different parts interact. An architectural configuration consists of independent components (software (including libraries) or hardware), interfaces of the components and connectors, which connect the components via their interfaces.***

Design definition: ***In contrast to the architecture, the design is concerned with the details inside one component.***

- [ ] General management
- [ ] Project management
- [ ] Requirements specification
- [ ] Architecture definition
- [ ] Design definition
- [ ] Software implementation
- [ ] Testing
- [ ] Customer support
- [ ] Safety management/assessment
- [ ] Quality management

Other:

7. Which safety standard(s) do you apply?

- [ ] None

a. Automotive

- [ ] ISO 26262
- [ ] IEC 61508
- [ ] ISO 25119
- [ ] ISO 15998
- [ ] NASA Safety Critical Guidelines (NASA-DB-1040)

Other:

b. Rail

- [ ] EN50126
- [ ] EN50128
- [ ] EN50129
- [ ] ISO 15998
- [ ] NASA Safety Critical Guidelines (NASA-DB-1040)

Other:

c. Avionics

- [ ] DO-178B/ ED-12B
- [ ] DO-178C/ED-12C
- [ ] DO254
- [ ] DO330

- [ ] NASA Safety Critical Guidelines (NASA-DB-1040)

Other:

d. Healthcare

- [ ] IEC62304
- [ ] IEC62061
- [ ] IEC 60601

Other:

e. Others

Other:

Back Next

Contact persons:
Grischa Liebel (Chalmers University of Technology), grischa@chalmers.se
Daniel Sauter (Itk Engineering) Daniel.Sauter@itk-engineering.de
Jos Langen (Verum), jos.langen@verum.com
Nadja Marko (VIRTUAL VEHICLE Research Center), Nadja.Marko@v2c2.at

40% completed

Model-based engineering

8. Which of the following answers conform to your view of model-based engineering?

Model-based engineering...

- [ ] is a new software and systems development paradigm.
- [ ] is the application of visual modeling.
- [ ] is in contrast to document-based, code-centric engineering.
- [ ] uses models for all different artifacts of the development process.
- [ ] makes use of problem-oriented structures.
- [ ] is a technique to speed up the design process.
- [ ] is a method to reduce the probability of systematic error.
- [ ] allows for fast formal and simulative validation.
- [ ] is designing a solution in tight loop with a virtual environment.
- [ ] places models in the center of the development process.

Other:

9. Please rate your experience with model-based engineering.

- ○ Newbie (< 1 year)
- ○ Moderate experience (1-3 years)
- ○ Highly experienced (> 3 years)

10. When did your division/department start applying model-based engineering?

- ○ 10 or more years ago
- ○ 1 – 5 years ago
- ○ 5 – 10 years ago
- ○ 1 month – 1 year ago
- ○ We haven't applied model-based engineering, yet.
- ○ I don't know

11. When did you personally use model-based engineering the last time?

- ○ 10 or more years ago
- ○ 1 – 5 years ago

- ○ 5 – 10 years ago
- ○ 1 month – 1 year ago
- ○ I am still using it
- ○ I have never used model-based engineering

12. Do you see yourself as a promoter or opponent of model-based engineering?

- ○ Promoter
- ○ Opponent
- ○ Neutral

13. What is the product you are targeting with model-based development?

- ○ Research demonstrator
- ○ Commercial product in small scale production (< 10 pieces)
- ○ Commercial product in medium scale production (10 .. 1000 pieces)
- ○ Commercial product in large scale series production (> 1000 pieces)
- ○ Non-commercial product for in-house use (e.g. for test of another product)
- ○ Other: [ ]

14. How relevant were the following reasons for introducing model-based engineering in your division/department?

	Not relevant			Relevant		Very relevant	I don't know
Need for quality improvements	○	○	○	○	○		○
Need for shorter development time	○	○	○	○	○		○
Need for cost savings	○	○	○	○	○		○
Required by customers	○	○	○	○	○		○
Required by standards	○	○	○	○	○		○
Need for formal methods	○	○	○	○	○		○
Need for traceability	○	○	○	○	○		○
Need to improve reliability	○	○	○	○	○		○
Need to improve availability	○	○	○	○	○		○
Need to improve safety	○	○	○	○	○		○
Need to improve integrity	○	○	○	○	○		○
Need to improve maintainability	○	○	○	○	○		○
Need to improve confidentiality	○	○	○	○	○		○
Need to improve reusability	○	○	○	○	○		○

Additional reasons and their relevance:

15. What were the effects of introducing model-based engineering in your division/department?

	highly negative		no effect		highly positive	I don't know
Quality	☐	☐	☐	☐	☐	☐
Development time	☐	☐	☐	☐	☐	☐
Efficiency of resulting code	☐	☐	☐	☐	☐	☐
Cost	☐	☐	☐	☐	☐	☐
Formal method adoption	☐	☐	☐	☐	☐	☐
Standard conformity	☐	☐	☐	☐	☐	☐
Traceability	☐	☐	☐	☐	☐	☐
Reliability	☐	☐	☐	☐	☐	☐
Availability	☐	☐	☐	☐	☐	☐
Safety	☐	☐	☐	☐	☐	☐
Integrity	☐	☐	☐	☐	☐	☐
Maintainability	☐	☐	☐	☐	☐	☐
Confidentiality	☐	☐	☐	☐	☐	☐
Reusability	☐	☐	☐	☐	☐	☐

Additional effects and their magnitude:

Back Next

Contact persons:
Grischa Liebel (Chalmers University of Technology), grischa@chalmers.se
Daniel Sauter (Itk Engineering) Daniel.Sauter@itk-engineering.de
Jos Langen (Verum), jos.langen@verum.com
Nadja Marko (VIRTUAL VEHICLE Research Center), Nadja.Marko@v2c2.at

60% completed

Model-based engineering - Part II

16. Which modeling environment do you use personally and which one is used in your division/department?

	Personal use	Division /Department
None	☐	☐
Rational Rhapsody	☐	☐
Artisan Studio	☐	☐
Enterprise Architect	☐	☐
Matlab / Simulink / Stateflow	☐	☐
UPPAAL	☐	☐
SCADE	☐	☐
ASCET	☐	☐
ASD:Suite	☐	☐
Labview	☐	☐
DYMOLA	☐	☐
SimulationX	☐	☐
Papyrus	☐	☐
Eclipse-based tools	☐	☐
In-house tool	☐	☐

Additional modeling environments for personal or division/department use:

17. Which modeling language(s) do you use personally and which one(s) are used in your division/department?

	Personal	Division /Department
None	☐	☐
UML	☐	☐
SysML	☐	☐
Modelica	☐	☐
Altarica	☐	☐

MARTE	☐	☐
Standard Domain-specific language/profile (e.g. EAST-ADL, AADL):	☐	☐
Company-internal approach/language:	☐	☐

Additional modeling languages for personal or division/department use:

18. Which type(s) of notations do you use personally and which one(s) are used in your division/department?

	Personal	Division /Department
None	☐	☐
Sequence based models (e.g., sequence diagrams, live sequence charts)	☐	☐
Structural models (e.g., class diagrams, component diagrams)	☐	☐
Finite State Machines (e.g., Statecharts, Stateflow)	☐	☐
Timed automata (or other state machines enriched with time information)	☐	☐
Block diagrams	☐	☐
Petri Nets	☐	☐
Hybrid Automata (or other state machines enriched with differential equations)	☐	☐
Differential Equations	☐	☐

Additional types of notations for personal or division/department use:

19. For which functional aspects of the system do you already use models personally and within your division/department?

	Personal	Division /Department
None	☐	☐
Structure (e.g., classes, components, blocks)	☐	☐
Static interfaces (e.g., signatures of method, messages, or signals)	☐	☐
Behavioral interfaces (e.g., allowed sequences of message)	☐	☐
Timing constraints	☐	☐
Specific scenarios or use cases of what the system must or must not do	☐	☐
Discrete State/Event Based specification (e.g., state/mode changes)	☐	☐
Continuous behavior (e.g., feedback controllers)	☐	☐
Hybrid behavior (combination of the two above, e.g., mode changes of feedback controllers)	☐	☐

Safety aspects	☐	☐

Additional functional aspects models are used for personally or within division/department:

20. How would you compare your usage and the usage within your division/department of model-based and non model-based tools for performing engineering activities?

	Personal	Division /Department
I/We do not perform engineering activities	☐	☐
I/We do not use model-based engineering tools	☐	☐
I/We use model-based engineering tools less than non model-based engineering tools	☐	☐
I/We use model-based engineering tools more than non model-based engineering tools	☐	☐
I/We use only model-based engineering tools	☐	☐

21. To what extent do the following potential shortcomings apply to the applied modeling approach?

	Does not apply at all		Partly applies		Fully applies	I don't know
Benefits require high efforts	☐	☐	☐	☐	☐	☐
High overhead involved	☐	☐	☐	☐	☐	☐
Many usability issues with the tools	☐	☐	☐	☐	☐	☐
Impossible/difficult to customize tools	☐	☐	☐	☐	☐	☐
Many limitations or difficulties on what can be expressed within the tools	☐	☐	☐	☐	☐	☐
Lack of proper semantics	☐	☐	☐	☐	☐	☐
Lack of completeness and consistency checks	☐	☐	☐	☐	☐	☐
Lack of model checking capabilities	☐	☐	☐	☐	☐	☐
Difficulties with integration into development process/current way of work	☐	☐	☐	☐	☐	☐
Difficulties with interfaces to interoperate with other tools (e.g. model exchange)	☐	☐	☐	☐	☐	☐
Difficulties of syntactic integration (e.g., different modeling language with different formats) with other tools	☐	☐	☐	☐	☐	☐
Difficulties of semantic interoperability (e.g., different definitions for the semantic of the modeling language) with other tools	☐	☐	☐	☐	☐	☐
Difficulties for distributed development	☐	☐	☐	☐	☐	☐
Difficulties/lack of traceability support	☐	☐	☐	☐	☐	☐
Difficulties with code generation capabilities	☐	☐	☐	☐	☐	☐

Difficulties of integration with existing legacy code	☐	☐	☐	☐	☐	☐
Difficulties with variability management support	☐	☐	☐	☐	☐	☐
Difficulties with version management support	☐	☐	☐	☐	☐	☐
High effort for training of developers	☐	☐	☐	☐	☐	☐

Additional shortcomings and their magnitude:

22. For which purpose does your division/department currently use models and what do you personally think models should be used for?

	Current use	Desired use
Not used	☐	☐
Information/documentation	☐	☐
Simulation	☐	☐
Code generation	☐	☐
Test-case generation	☐	☐
Structural consistency checks	☐	☐
Behavioral consistency checks	☐	☐
Formal verification/theorem proving	☐	☐
Safety compliance checks	☐	☐
Timing analysis	☐	☐
Reliability analysis	☐	☐
Traceability	☐	☐

Additional current or future model usages:

23. Which types of integration mechanisms do you currently use and which ones should be used in the future?

	Current use	Desired use
No automatic integration	☐	☐
(Hyper-) Links	☐	☐
Common database/model accessed by different tools	☐	☐
Export/Import via defined file formats	☐	☐
Tool adapters/plug-ins	☐	☐
Message passing between tools	☐	☐

Additional types of integration mechanisms which are or should be used:

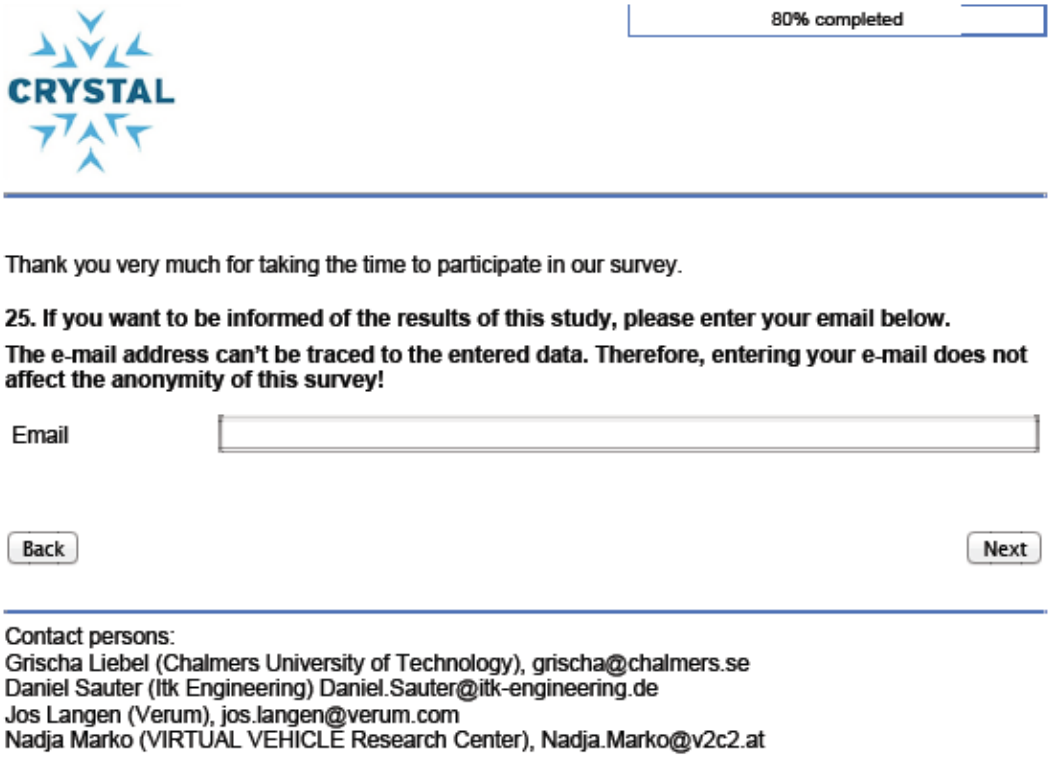
24. In which phases of the development process are you using model-based engineering?

- [ ] Requirements Analysis
- [ ] System Architecture
- [ ] Subsystem/Component Design
- [ ] Implementation
- [ ] Subsystem/Component Test
- [ ] Integration Test
- [ ] System Test

Other:

Back Next

Contact persons:
Grischa Liebel (Chalmers University of Technology), grischa@chalmers.se
Daniel Sauter (Itk Engineering) Daniel.Sauter@itk-engineering.de
Jos Langen (Verum), jos.langen@verum.com
Nadja Marko (VIRTUAL VEHICLE Research Center), Nadja.Marko@v2c2.at



80% completed

Thank you very much for taking the time to participate in our survey.

25. If you want to be informed of the results of this study, please enter your email below.

The e-mail address can't be traced to the entered data. Therefore, entering your e-mail does not affect the anonymity of this survey!

Email

Back Next

Contact persons:
Grischa Liebel (Chalmers University of Technology), grischa@chalmers.se
Daniel Sauter (Itk Engineering) Daniel.Sauter@itk-engineering.de
Jos Langen (Verum), jos.langen@verum.com
Nadja Marko (VIRTUAL VEHICLE Research Center), Nadja.Marko@v2c2.at