 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 Validation Models - V1

D612.011

DOCUMENT INFORMATION

Project	CRYSTAL
Grant Agreement No.	ARTEMIS-2012-1-332830
Deliverable Title	Specification, Development and Assessment for Validation Models - V1
Deliverable No.	D612.011
Dissemination Level	CO
Nature	R
Document Version	V3.00 (final version)
Date	2014-01-29
Contact	Gregorio Barberio
Organization	MATE
Phone	
E-Mail	g.barberio@mateconsulting.it

AUTHORS TABLE

Name	Company	E-Mail
Aniello Amato	Mate Consulting srl	a.amato@mateconsulting.it
Barberio Gregorio	Mate Consulting srl	g.barberio@mateconsulting.it
Stieglbauer Gerald	AVL/GRZ	Gerald.Stieglbauer@avl.com
Wallner Alfred	AVL/GRZ	alfred.wallner@avl.com
Settelmeier Joerg	AVL/DE	joerg.settelmeier@avl.com
Serrie Chapman	Infineon	serrie.chapman@infineon.com
Galpin Darren	Infineon	darren.galpin@infineon.com;
Tichy Matthias	CTH	tichy@chalmers.se

REVIEW TABLE

Version	Date	Reviewer
Internal Review	2013-12-03	Gregorio Barberio
Internal Review	2014-01-13	Valeria Vittorini
External Review	2014-01-27	Vasaiely, Parham
External Review	2014-01-29	Renato De Guglielmo

CHANGE HISTORY

Version	Date	Reason for Change	Pages Affected
V1.00	2013-12-03	First issue	
V2.00	2014-01-15	Minor changes after internal review	
V3.00	2014-01-29	Some minor changes after external review: - updated the references table - deleted empty tables for technical items	

CONTENT

Content of Tables

Content of Figures

1 Introduction

1.1 Role of deliverable

This document aims to show the results of the first requirement collection phase for work package 6_12. Since this deliverable is due at an 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.

In this deliverable, Bricks are documented which are developed in the WP6.12. This deliverable is updated iteratively, three times during the project runtime, based on the corresponding milestones of the project. Therefore, the Brick documentations in this document represent an evolutionary process of continuous development and enhancement of the CRYSTAL solutions, and complementary to previous version of this deliverable.

Here a brief introduction to the WP 6.12 scope.

To validate a complex industrial systems, starting from the system requirements, test scenarios should be defined by V&V team (independent from the development team), usually using a model describing the system itself. This model often does not allow any automatic verification of its feasibility. Any change of the requirements implies the manual identification of the tests impacted by this change and then a modification of the tests themselves.

Moreover, it is not possible to define, automatically, system tests from the model itself, but this definition of test cases is made manually, starting from a model that is only a representation of the system behaviour.

To reduce the effort related to these activities it is necessary to improve the integration between the different steps of the V&V process.

The objectives should be summarized as:

- model complex and asynchronous systems of different domains in the CRYSTAL scope;
- limit efforts for test case generation;
- reduce time needed to modify test cases after changes in requirements;
- support users during the analysis phase of the entire system life cycle.

1.2 Relationship to other CRYSTAL Documents

This table shows the relationship between other CRYSTAL documents and bricks of this deliverable.

CRYSTAL		
Code	**Title**	**Bricks**
D301.010	Use Case Definition	B3.12
D304.011	Use Case Definition.	B3.12 B3.100
D307.011	Public Use Case Automotive.	B3.100
D501.010	Data and methodologies report.	B5.1 B5.3 B5.4
D501.020	Use Requirements Specification.	B5.1 B5.3 B5.4
D612.011 Specification, Development and Assessment for Validation Models - V1		

Table 1-1: Relationship to other Crystal Documents

1.3 Structure of this document

This document is organized as follows:

- Chapter 1 (this chapter) provides an overview of the document;
- Chapters 2-13 are arranged by bricks, each brick is represented in a separate section and each section contains at least a brief description of the brick.

  A brick in Crystal may be a Software Tool (Single Tool, Pre-Integrated Set of Tools, Software Platform), a Method (how to…), a Process (work flows, information flows,…) or a Specification (Language, Interoperability,…).

  For each bricks of WP 6.12 there is a base common structure (other sub-chapter are available as needed) as follow:

  - Description, a general description of the purpose and functionality;
  - Use case coverage and application;
  - General improvement;
  - Integration and interoperability.

  Extensions, enhancements, improvements of Bricks developed in CRYSTAL to meet the UC needs are technical solutions represented and documented as Technical Items (TI) A Brick can be associated by one or more Technical Items.
- The remaining chapters contains tables about Abbreviations, References, etc.

2 Validation Process (B3.12)

2.1 Description

Name:	Validation Process
Contact:	TICHY Matthias (tichy@chalmers.se)
Dependencies	Brick B3.7 Model-based requirements engineering
License	N/A
Additional information	

This methodology brick contains the definition of a validation process for automotive systems with a specific focus on the Use Case 3.1. The validation process targets the model-driven software engineering approach and additionally, as a result of the Use Case 3.1 definition, focuses on the requirements and design phases where modeling will be used. This brick also depends on the Brick B3.7 “Model-based requirements engineering” that focuses on using model-based approaches for formally specifying requirements for embedded systems.

In the initial phase, we worked on the requirements engineering phase and performed a systematic mapping study to assess the state of the art of validation in model-based requirements engineering for automotive systems together with Brick B3.7. A specific focus of the research was to identify approaches that have been empirically validated in industrial projects, as those are the most relevant for Crystal.

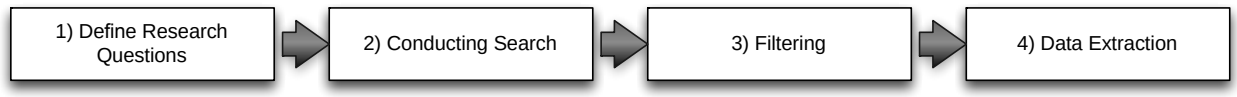


Figure 2-1: Research method

Figure 2 1 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 extracted 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 validation and verification activities are supported?

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 on all papers that 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 the 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. 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 3). We refer to Deliverable D603.011 for a description of which domains the different papers target, which modeling languages are used in the different papers, and which aspects of the system’s requirements can be modeled.

V & V technique	All papers	Papers reporting on industrial application
test case generation	17	[S27], [S28], [S2], [S21], [S19], [S4]
consistency verification	21	[S18], [S13], [S28], [S30], [S17], [S8], [S12]
model checking	13	[S20], [S25], [S21], [S13], [S6]
simulation	12	[S20], [S15], [S9], [S17], [S13]
formal verification in general	12	[S30], [S10]
other	17	[S1], [S23]

Figure 2-2: Most popular validation and verification techniques

From our initial results, 20% of the papers 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. From a validation perspective, many different validation techniques were employed. However, clearly,

both in all papers and in the ones with industrial application, four techniques are employed most: static consistency checks, test case generation, simulation, and model checking as formal method.

2.2 Identified requirements for extension

Based on these results and discussions in the Use Case 3.1, we currently focus on the requirements phase and use the Brick B3.7 on scenario-based modelling of requirements, specifically Modal Sequence Diagrams (See Deliverable D603.011) as modeling approach. With respect to validation, we currently focus on simulation from the list above. The employed approach/tool supports manual simulation by stepping through the scenarios and their interaction.

One identified requirement is the support for using the scenarios as test oracles in automated tests of systems that implement the requirements. This requires an automation of the simulation and the definition of an appropriate interface to interact with the test oracle.

Another requirement for extension is the support for the integration of continuous behaviour. While the currently employed modeling formalism (and the simulation) supports scenarios of discrete message exchange and corresponding manual simulation, it was identified that continuous behaviour like the speed and the acceleration of the car is an important part of the requirements and will be integrated into the formalism in Brick B3.7. This has the obvious impact on the validation support such that it also must consider that part to the required extent.

2.3 Use Case coverage and application

The brick is used in the Use Case 3.1. Furthermore, due to the dependency on Brick B6.3 and its involvement in Use Case 3.4, we get specific needs from that use case as well.

While the exact needs from Use Case 3.1 need to be defined, the above described identified requirements coming from the Use Case 3.4 are also suitable for Use Case 3.1.

2.4 General Improvement

The definition of Technical Items, which represent extensions, enhancements and improvements of Bricks developed in CRYSTAL to meet the UC needs, except from Interoperability, will be part of the next phase of the project.

2.5 Integration and Interoperability

The definition of Technical Items, which represent solutions related to the interoperability of Bricks, will be part of the next phase of the project.

3 AVL VeVaT/Magic (B3.49)

3.1 Description

Name:	AVL VeVaT/Magic
Contact:	STIEGLBAUER Gerald (gerald.stieglbauer@avl.com)
Technical Information: This brick will be about requirement verification and validation based on measurement results of particular test cycles in the automotive domain.	
Operation System/platform	Windows
Version	N/A
Type of Input Data	Formal Requirements, Measurement results
Type of Output Data	Verified Requirements
Dependencies	N/A
License	
Additional information	N/A

AVL MAGIC automates the post processing. It is designed to manipulate large amount of data. The extensive toolkit library makes the design of processing sequence straight forward. Fully integrated in the automation system it enables a direct online processing while the tests are running.

AVL VeVaT works on top of Magic and is used for verification and validation based on the results delivered by Magic and on the basis of the associated requirements.

3.1.1 Tool/method description: what will you find in AVL VeVaT/Magic

AVL VEVAT is a **Ve**rification & **Va**lidation **T**ool for various software products resp. software modules which generate voluminous numerical output – e.g. numerous single result values and/or huge sequences of numerical values (e.g. multi-channel time history data).

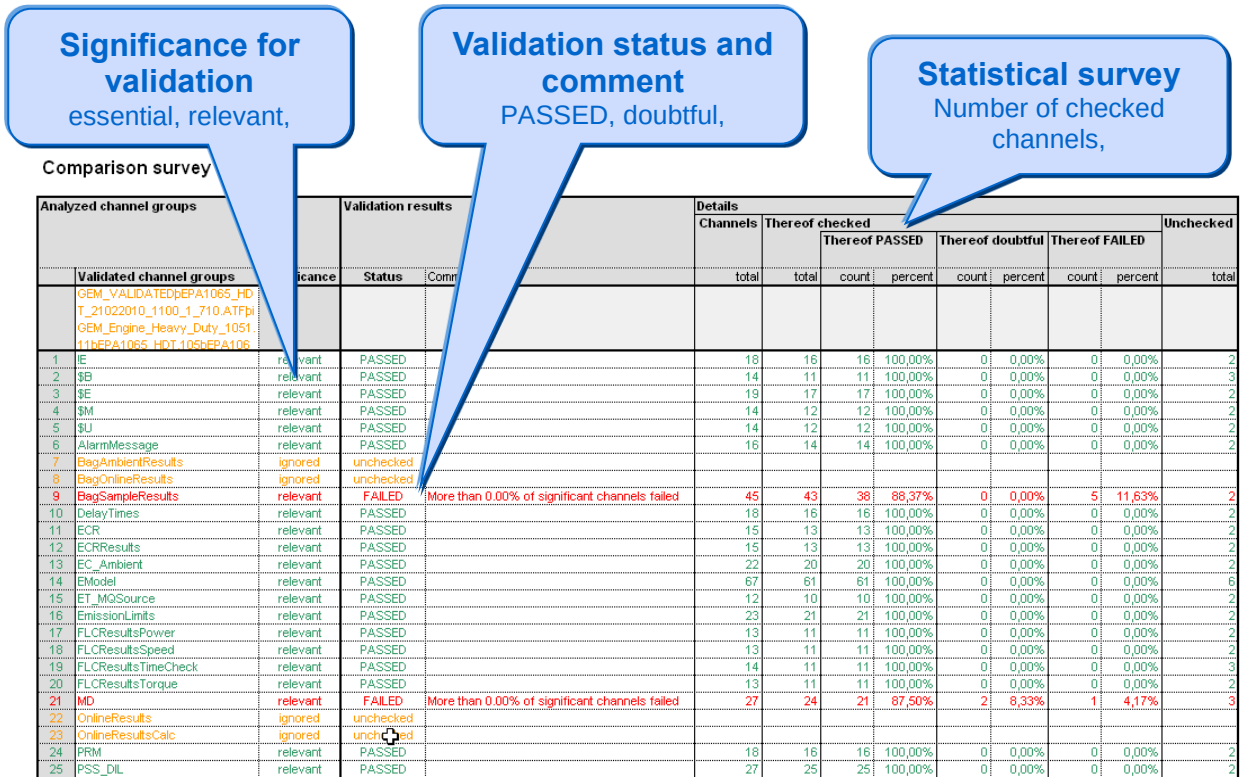
It provides two approaches for validating product functionality via checking the correctness of its numerical output data:

- Comparison of actual output data to already validated reference output data (mainly used for regression tests, where output data of a new product version get compared to validated data of a preceding software release)
- Detecting/deriving significant properties of output data (time history data) and compare them to numerical product requirements (e.g. deriving properties of a vehicle braking event from road measurement data and checking, whether e.g. braking time, braking distance, deceleration, ABS-influences etc. reside within required limits); therefore no pre-validated reference data but just the numerical requirements are needed.

In order to obtain practical validation results (PASSED / FAILED statements), comparison parameters (significance of checked values, acceptable deviations resp. tolerances etc.) can be optionally user-defined at several hierarchy levels (for entire result files, for particular data channel groups, or even for individual data channels resp. single values).

VEVAT generates validation reports, which supply PASSED / DOUBTFUL / FAILED validation statements at several hierarchy levels, i.e. an overall statement and more detailed statements for subsections of analyzed data – even for single values where beside PASSED / FAILED statements furthermore the reference value, the actual value, the calculated deviation etc. is displayed.

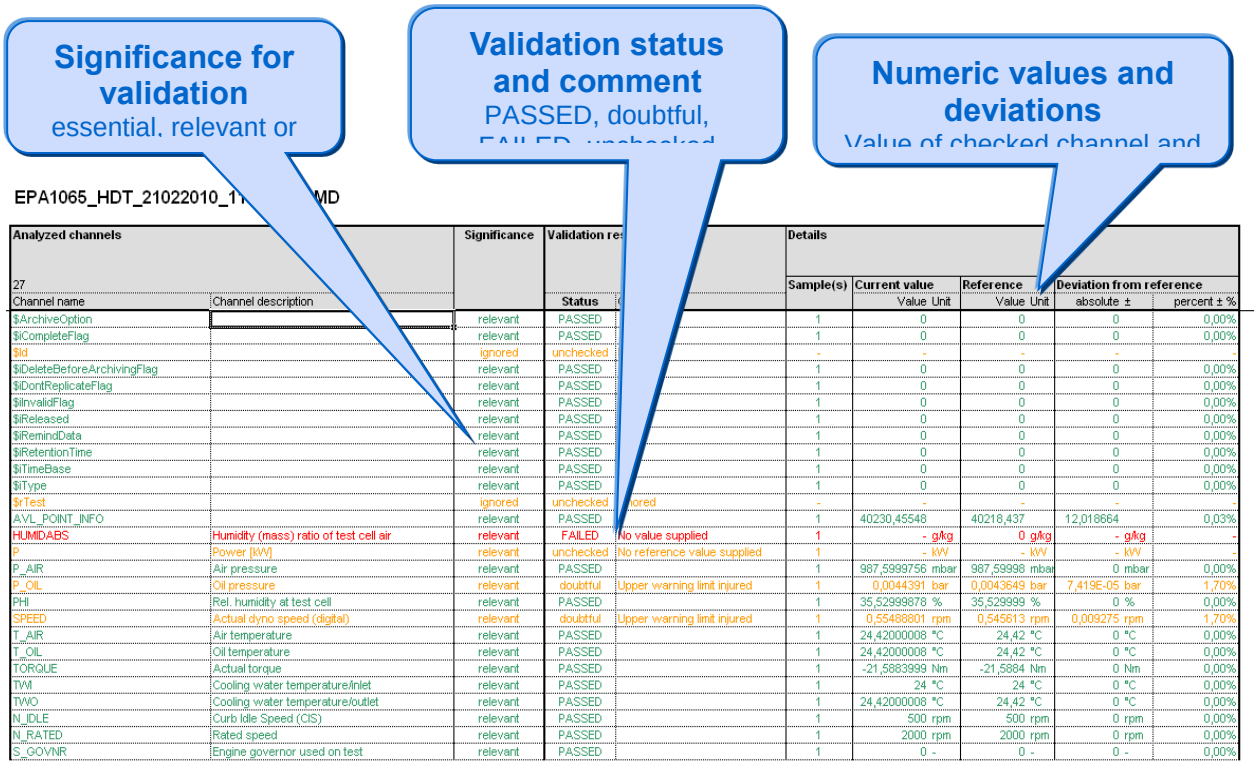
The following example shows a validation report of an emission test result at channel group level (number and percentage of channels that are stated as passed, failed, unchecked etc.):



	Analyzed channel groups		Validation results		Details								
					Channels	Thereof checked							Unchecked
							Thereof PASSED		Thereof doubtful		Thereof FAILED		
	Validated channel groups	icance	Status	Comr	total	total	count	percent	count	percent	count	percent	total
	GEM_VALIDATEDbEPA1065_HDT_21022010_1100_1_710.ATFbi GEM_Engine_Heavy_Duty_1051 11bEPA1065_HDT 105bEPA106												
1	IE	relevant	PASSED		18	16	16	100,00%	0	0,00%	0	0,00%	2
2	$B	relevant	PASSED		14	11	11	100,00%	0	0,00%	0	0,00%	3
3	$E	relevant	PASSED		19	17	17	100,00%	0	0,00%	0	0,00%	2
4	$M	relevant	PASSED		14	12	12	100,00%	0	0,00%	0	0,00%	2
5	$U	relevant	PASSED		14	12	12	100,00%	0	0,00%	0	0,00%	2
6	AlarmMessage	relevant	PASSED		16	14	14	100,00%	0	0,00%	0	0,00%	2
7	BagAmbientResults	ignored	unchecked										
8	BagOnlineResults	ignored	unchecked										
9	BagSampleResults	relevant	FAILED	More than 0.00% of significant channels failed	45	43	38	88,37%	0	0,00%	5	11,63%	2
10	DelayTimes	relevant	PASSED		18	16	16	100,00%	0	0,00%	0	0,00%	2
11	ECR	relevant	PASSED		15	13	13	100,00%	0	0,00%	0	0,00%	2
12	ECRResults	relevant	PASSED		15	13	13	100,00%	0	0,00%	0	0,00%	2
13	EC_Ambient	relevant	PASSED		22	20	20	100,00%	0	0,00%	0	0,00%	2
14	EModel	relevant	PASSED		67	61	61	100,00%	0	0,00%	0	0,00%	6
15	ET_MQSource	relevant	PASSED		12	10	10	100,00%	0	0,00%	0	0,00%	2
16	EmissionLimits	relevant	PASSED		23	21	21	100,00%	0	0,00%	0	0,00%	2
17	FLCResultsPower	relevant	PASSED		13	11	11	100,00%	0	0,00%	0	0,00%	2
18	FLCResultsSpeed	relevant	PASSED		13	11	11	100,00%	0	0,00%	0	0,00%	2
19	FLCResultsTimeCheck	relevant	PASSED		14	11	11	100,00%	0	0,00%	0	0,00%	3
20	FLCResultsTorque	relevant	PASSED		13	11	11	100,00%	0	0,00%	0	0,00%	2
21	MD	relevant	FAILED	More than 0.00% of significant channels failed	27	24	21	87,50%	2	8,33%	1	4,17%	3
22	OnlineResults	ignored	unchecked										
23	OnlineResultsCalc	ignored	unchecked										
24	PRM	relevant	PASSED		18	16	16	100,00%	0	0,00%	0	0,00%	2
25	PSS_DIL	relevant	PASSED		27	25	25	100,00%	0	0,00%	0	0,00%	2

Figure 3-1: VEVAT Validation report on channel group level

The following example shows a validation report of an emission test result at single channel resp. single value level (PASSED, FAILED, unchecked statement as well as numerical values and deviations):



EPA1065_HDT_21022010_1 MD

Analyzed channels		Significance	Validation re		Details				
27					Sample(s)	Current value	Reference	Deviation from reference	
Channel name	Channel description		Status			Value Unit	Value Unit	absolute ±	percent ± %
$ArchiveOption		relevant	PASSED		1	0	0	0	0,00%
$iCompleteFlag		relevant	PASSED		1	0	0	0	0,00%
$id		ignored	unchecked		-	-	-	-	-
$iDeleteBeforeArchivingFlag		relevant	PASSED		1	0	0	0	0,00%
$iDontReplicateFlag		relevant	PASSED		1	0	0	0	0,00%
$iInvalidFlag		relevant	PASSED		1	0	0	0	0,00%
$iReleased		relevant	PASSED		1	0	0	0	0,00%
$iRemindData		relevant	PASSED		1	0	0	0	0,00%
$iRetentionTime		relevant	PASSED		1	0	0	0	0,00%
$iTimeBase		relevant	PASSED		1	0	0	0	0,00%
$iType		relevant	PASSED		1	0	0	0	0,00%
$rTest		ignored	unchecked	ored	-	-	-	-	-
AVL_POINT_INFO		relevant	PASSED		1	40230,45548	40218,437	12,018664	0,03%
HUMIDABS	Humidity (mass) ratio of test cell air	relevant	FAILED	No value supplied	1	- g/kg	0 g/kg	- g/kg	-
P	Power [kW]	relevant	unchecked	No reference value supplied	1	- kW	- kW	- kW	-
P_AIR	Air pressure	relevant	PASSED		1	987,5999756 mbar	987,59998 mbar	0 mbar	0,00%
P_OIL	Oil pressure	relevant	doubtful	Upper warning limit injured	1	0,0044391 bar	0,0043649 bar	7,419E-05 bar	1,70%
PHI	Rel. humidity at test cell	relevant	PASSED		1	35,52999878 %	35,529999 %	0 %	0,00%
SPEED	Actual dyno speed (digital)	relevant	doubtful	Upper warning limit injured	1	0,55488801 rpm	0,545613 rpm	0,009275 rpm	1,70%
T_AIR	Air temperature	relevant	PASSED		1	24,42000008 °C	24,42 °C	0 °C	0,00%
T_OIL	Oil temperature	relevant	PASSED		1	24,42000008 °C	24,42 °C	0 °C	0,00%
TORQUE	Actual torque	relevant	PASSED		1	-21,5883999 Nm	-21,5884 Nm	0 Nm	0,00%
TWI	Cooling water temperature/inlet	relevant	PASSED		1	24 °C	24 °C	0 °C	0,00%
TWO	Cooling water temperature/outlet	relevant	PASSED		1	24,42000008 °C	24,42 °C	0 °C	0,00%
N_IDLE	Curb Idle Speed (CIS)	relevant	PASSED		1	500 rpm	500 rpm	0 rpm	0,00%
N_RATED	Rated speed	relevant	PASSED		1	2000 rpm	2000 rpm	0 rpm	0,00%
S_GOVNR	Engine governor used on test	relevant	PASSED		1	0 -	0 -	0 -	0,00%

Figure 3-2: VEVAT Validation report on single value level

AVL VeVaT works on top of the tool AVL Magic and is used for verification and validation based on the results generated at test runs and on the basis of the associated requirements.

AVL MAGIC automates the post processing. It is designed to manipulate large amount of data. The extensive toolkit library makes the design of processing sequence straight forward. Fully integrated in the automation system it enables a direct online processing while the tests are running.

3.2 Use Case coverage and application

AVL VeVaT/Magic will be mostly covered by WP3.4 (UC3.4a). Here it has mostly deal with two input data categories: Requirements and measurement results of test cycle iterations in the automotive domain.

Requirements for vehicle development and testing are represented in different forms. At the beginning of a project, they are formulated by informal textual representations. In order to verify requirements (semi-) automatically the need to be (semi-) formalized. (Semi-) formalized representations would be for instance boilerplate expressions.

In the corresponding use case scenario of WP3.4, for instance, the WLTP specification (which is an up-coming specification for world-wide harmonized emission legislation) is the foundation for the testing requirements (in addition to the vehicle requirements). Thus these specifications have to be formalized. In our use case we will examine three (semi-)formal representations: *Boilerplates*, *requirement models* in form of SysML models and *sequenced based requirement* representations (based on SysML as well). All these representations are created by different tools and need to be interlinked to enable traceability.

The requirements should be represented in as much representations as possible/useful. The following tools are considered for the various representations:

- HP Quality Center (informal requirement representation, an ALM tool such as PTC Integrity is considered to be an alternative tool)
- ViF Boilerplate Prototype tool
- Artisan Studio (for SysML representations)

Measurement results is the data output of a concrete test-run execution. These results need to be analysed and requirements have to be validated against these analyses. Furthermore test-run inputs and measurement outputs are building the basis for the definition of a calibration model.

The following requirements could be associated with concrete measurement results. The exact content and meaning of measurement results often differs from case to case and especially between different testing phases. Consequently a mapping and unique interpretation of the measurement results is considered as an interoperability challenge. The table below associates selected vehicle and testing requirements to particular tools that manages them in various testing phases. If a 1:1 mapping of measurement result and the related requirement is not possible measurement result post-processing is provided by separate tools such as AVL Magic.

In WP3.4 there, a well specified engineering method deals with the task of verification of requirements using AVL VeVaT/Magic. Purpose of this engineering method is the verification of requirements against a specific *test results* (e.g. *measurement results*) of a specific *test case*. At this point of the engineering method, it is not distinguished between natural textual requirements and (semi-)formalized requirements. Thus the tools on the requirement management side are *HP Quality Center*, *Artisan Studio* and the *boilerplate prototype application*. These requirements are verified by concrete test cases that lead to specific measurement results. *AVL Santorin* is the tool of choice in this use case to manage and access these measurement results. The actual process of requirement verification is captured by the AVL VeVaT tool. Due to the availability of proper data artefact links, AVL VeVaT is able to compare the measurement results against the formalized requirements and finally verifies or falsifies the associated natural language requirement. If some measurement value post-processing is needed for that step, AVL VeVaT instruments AVL Magic, which is designed for such tasks.

3.3 General Improvement

The major objective of this task is to improve interoperability with other tools which are usually used in tight collaboration with AVL VeVaT/Magic. A use case for such collaboration will be developed in WP3.4a. Special focus of collaboration will be the interoperability with the brick Simulation Model Backbone Database (B3.83), which will be developed in WP6.13. It is expected that for AVL VeVaT an increase amount of features has to be implemented for the use case – e.g. support for handling requirement lists (requirement definitions as well as requirement validation status) as shown in the picture below:

The definition of Technical Items, which represent extensions, enhancements and improvements of Bricks developed in CRYSTAL to meet the UC needs, except from Interoperability, will be part of the next phase of the project.

3.4 Integration and Interoperability

In order to improve collaboration as described above, AVL VeVaT/Magic should be seamlessly integrated into the CRYSTAL interoperability standard, which a special focus on interoperability with brick B3.83 (WP6.13) as well as with bricks that are used in WP3.4.

In addition to that Figure 3-3: Integration of AVL VeVaT/Magic within the IOS concept shows a possible concept for IOS/OSLC integration. The corresponding OSLC RM/QM domains provide high-level data structures for the definition of data relations. In case of an AVL VeVaT/Magic integration an adequate linked-data structure then provides relations between concrete requirements (in natural language or even semi-formalized), test cases and measurement values (which are created by the execution of test cases). In addition, AVL Santorin (applied within WP3.4) is a specialized tool for measurement result management. AVL Santorin adheres to the ASAM ODS standard created for a standardized representation measurement values. AVL Magic is able to interpret this standard and thus can import the measurement values for post-processing. On the one hand, AVL VeVat (which works on top of AVL Magic) directly communicates with AVL Magic regarding the results of this post-processing step. On the other hand it should have a corresponding access layer (via an OSLC adapter) to the OSLC-based data structure. In that way it can analyse the belonging semi-formalized requirements with the post-processed measurement data. Depending on the result of this analysis, corresponding flags about the status of the belonging requirements are set in the requirement management tools. These requirement management tools will be extended as well by a OSLC adapter in order to enable that kind of tool interoperability.

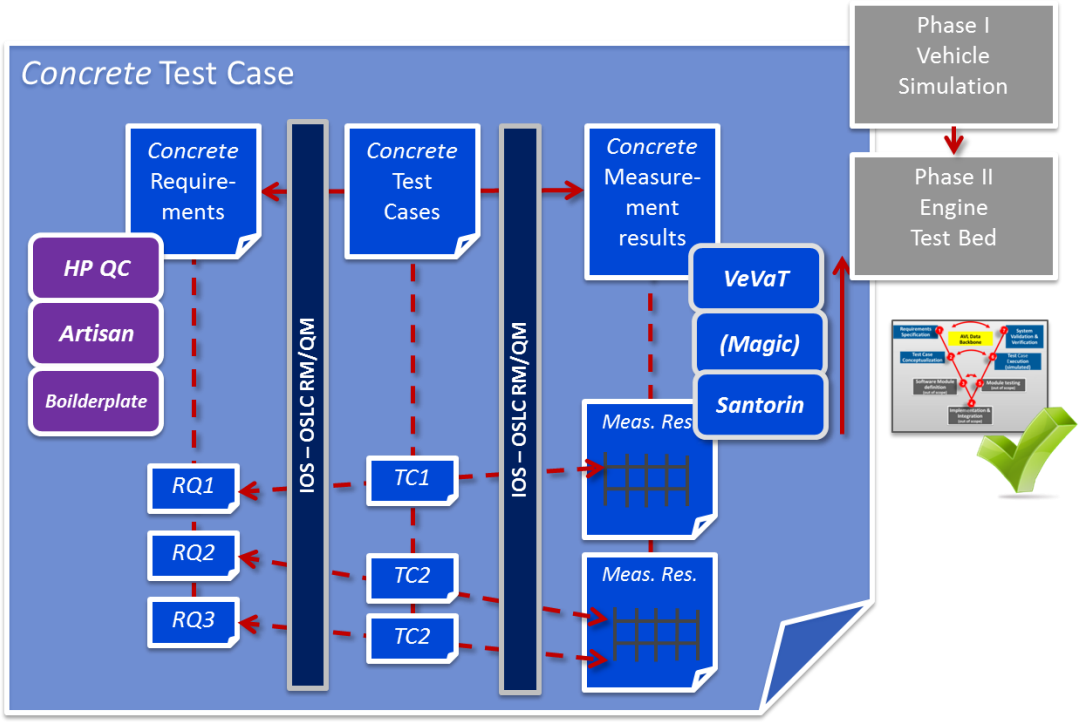


Figure 3-3: Integration of AVL VeVaT/Magic within the IOS concept

Requirement definitions					Validation results				
Requirement-ID	Description	Lower limit	Upper limit	Unit	Status	Comment	Detected minimum	Detected maximum	Unit
VV_RECUP_General_01-BP	The Hybrid system shall be able to check the HV SOC.				unchecked	No value supplied	-	-	
VV_RECUP_General_07-BP	If clutch pedal signal = 0 , the Hybrid system shall enable the recuperation.		0,00		PASSED		0	0	
VV_RECUP_General_08-BP	The Hybrid system shall enable the recuperation when the Driver activates the service brake.	1,00			FAILED	Lower error limit violated	0	1	
VV_RECUP_General_08-BP_	The Hybrid system shall enable the recuperation when vehicle in motoring mode.	1,00			PASSED		1	1	
VV_RECUP_General_10-BP	The Hybrid system shall keep the temperature of HV battery within 10 and 30 °C.	10,00	30,00	°C	PASSED		20,017292	20,1153238	°C
VV_RECUP_General_11-BP	The e-motor shall be able to generate the brake torque.				unchecked	No value supplied	-	-	
VV_RECUP_General_12-BP	The Hybrid system shall be able to conform to the NVH criteria.				unchecked	No value supplied	-	-	
VV_RECUP_General_14-BP	The Hybrid system shall enable the recuperation when vehicle speed greater than 60 km/h.	60,00		km/h	FAILED	Lower error limit violated	13,55733756	73,2275136	km/h
VV_RECUP_General_15-BP	While motoring_recuperation , the Hybrid system shall decelerate the vehicle less than 10 m/s².		10,00	m/s²	PASSED		-6,661513303	-1,364381317	m/s²
VV_RECUP_General_16-BP	The charging power of Hybrid system shall depend on the HV SOC.				unchecked	No value supplied	-	-	
VV_RECUP_General_18-BP	If temperature of e-motor greater than 70 °C the Hybrid system shall not charge the HV battery.		70,00	°C	PASSED		20,017292	20,1153238	°C
VV_RECUP_General_18-BP_	If temperature of HV battery greater than 50 °C the Hybrid system shall not charge the HV battery.		50,00	°C	PASSED		20,5443213	22,872681	°C
VV_RECUP_General_17-BP	The the Hybrid system shall maintain the HV SOC less than 95 percent.		95,00	percent	PASSED		89,052839	89,8554285	percent

Figure 3-4: VEVAT Internal requirements container

The left group of columns **Requirement definitions** contains the requirement properties that are supplied by the requirements management tool (HP Quality Center®) via IOS/OSLC. The right group of columns **Validation results** contains the validation status, comments etc. that can be returned to the requirements management tool via IOS/OSLC.

Besides of improvements on the tool that become necessary in order to comply with the CRYSTAL interoperability standard, several features have to be implemented for the tools in order to fit the related use case requirements.

The definition of Technical Items, which represent solutions related to the interoperability of Bricks, will be part of the next phase of the project.

4 RequisitePro (B3.86)

4.1 Description

Name:	RequisitePro
Contact:	Serrie.chapman@infineon.com
http://www-03.ibm.com/software/products/en/reqpro	
Operating System	Windows terminal server
Version	N/A
Type of Input Data	Manual semi-formalised natural language
Type of Output Data	ARQE.xml
Dependencies	Currently none – manual input. Possible link to a Quality tool (DODT from the ARTEMIS CESAR project) and B3.87 Clearquest
License	Token license
Additional information	Requisite Pro is being passed out currently by IBM so resources spent on integrating this tool wrt a long term solution may not be truly feasible and we may consider moving the work to integrating the replacement solution in its place

Requisite Pro is IBM's Requirements Management tool http://www-03.ibm.com/software/products/en/reqpro/ . In reality IBM are in fact phasing this out for a new replacement, however within inFineon, although we have analysed and decided to move from RequisitePro to Visure http://www.visuresolutions.co.uk , within the 40nm Crystal Infineon project it is too late to move over to the new tool so we need to consider how we can interface.

4.1.1 General Description

IBM® Rational® RequisitePro® is a requirements and use case management tool for project teams. Teams can author and share their requirements using familiar document-based methods, while using database capabilities such as traceability and impact analysis.

4.2 Use Case coverage and application

4.2.1 Use Case 3.3

Improve the capture, allocation and implementation of all stakeholder requirements for the next Infineon 40nm Microcontroller product family to:

- Improve comprehension and communication of requirements internally and externally.

4.2.2 Requirements fulfilled by initial tool/method version

Currently the requirements have been manually improved to be of good quality and to be an internal semi-formal Natural language type. The Hierarchy within the tool has been much improved since the last product family as it is well considered and can be used throughout the flow to assist with tooling.

Automation of documentation from the database has already been implemented and we are working on extractions for the Verification flow.

4.2.3 What will be implemented/provided in the CRYSTAL project

A full automation from the RequisitePro database into tooling and documentation will be achieved and will ensure data integrity within the flow

4.2.3.1 New and improved features

- Traceability between internal and external requirements to allow well separated documents for different customers to improve customer interface
- Automation of extract of atomic requirements to be used directly for specification and for the test plan tooling

4.3 General Improvement

4.3.1 Implementation

A Cron job delivers an updated xml within the configuration management system whenever and only when an update has occurred that affects the requirements

A Script is triggered by this which does an intelligent diff, it is needed to identify only the sub IP/domain xmls that need updating and overwrite the original xml within clearcase configuration management system.

The definition of Technical Items, which represent extensions, enhancements and improvements of Bricks developed in CRYSTAL to meet the UC needs, except from Interoperability, will be part of the next phase of the project.

4.4 Integration and Interoperability

4.4.1 Interoperability requirements

Linkage between the Clearquest change management system and RequisitePro database is currently an issue and may need a common interface – although a DODT equivalent may provide an interim step, in which case interoperability with the DODT will be necessary.

The definition of Technical Items, which represent solutions related to the interoperability of Bricks, will be part of the next phase of the project.

4.4.2 How will this brick be integrated in the UC

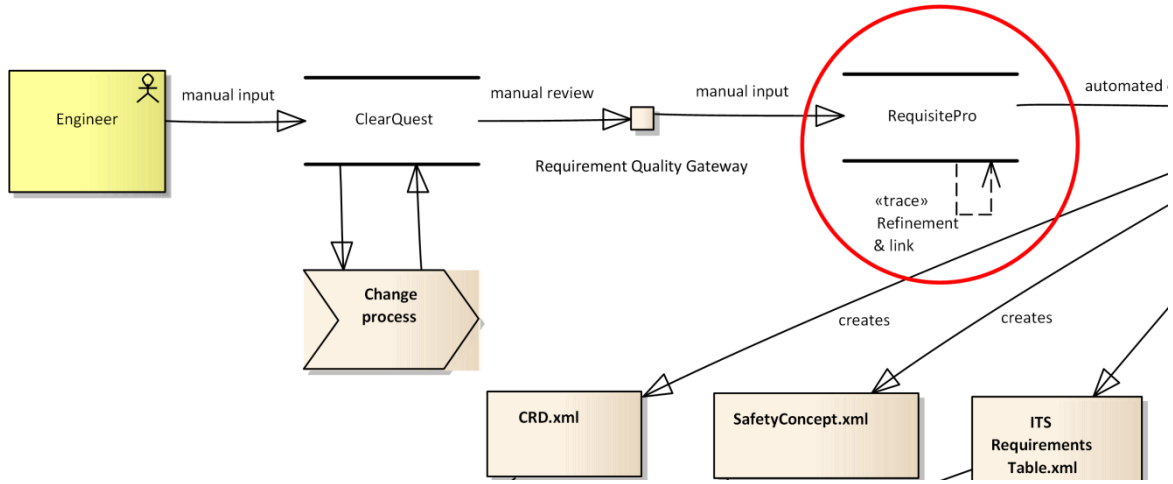


Figure 4-1: RequisitePro toolchain integration

5 ClearQuest (B3.87)

5.1 Description

Name:	ClearQuest
Contact:	Serrie.chapman@infineon.com
http://www-03.ibm.com/software/products/en/clearquest/	
Operating System	Windows terminal server
Version	N/A
Type of Input Data	Manual
Type of Output Data	Manual
Dependencies	None currently needs to link to RequisitePro B3.86
License	Rational Common Licensing multisite
Additional information	As with Requisite Pro, this is being replaced with Jira. Analysis of the cost of integrating an automated solution between Clearquest and Requisite pro may suggest that we look at the Jira/Visure integration instead

IBM® Rational® ClearQuest® is application lifecycle management (ALM) software that provides flexible change and defect tracking, customizable processes, real-time reporting and lifecycle traceability for better visibility and control of the software development lifecycle.

Essentially this is a database which contains, amongst other data, a list of all requirements that come into the project after the first requirements analysis and quality review of the agreed product requirements has occurred.

5.2 Use Case coverage and application

5.2.1 Requirements fulfilled by initial tool/method version

A clear set of rules and processes are already defined for the Change management tooling – although the translation process from the Change database into the requirement database is manual and there is a risk of data corruption between the two.

5.2.2 What will be implemented/provided in the CRYSTAL project

Automation between ClearQuest and RequisitePro to ensure quality and data integrity – via the DODT (see Brick B3.99) is the improvement that we plan to implement within the Crystal project.

5.3 General Improvement

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

5.4 Integration and Interoperability

5.4.1 How will this brick be integrated in the UC

After the initial collation and review of the requirements and once the product development has started ALL changes, new or rejected requirements must go through the Change management system, thus making this the starting Brick in the process. There needs to be a very specific set of rules & process relating to any changes after a requirement contractual agreement, this should be documented and under Configuration management.

All Changes that relate to a change in the requirements list must ensure that they meet the quality criteria – currently this is a manual process, within crystal we will attempt to automate this (see Brick B3.99).

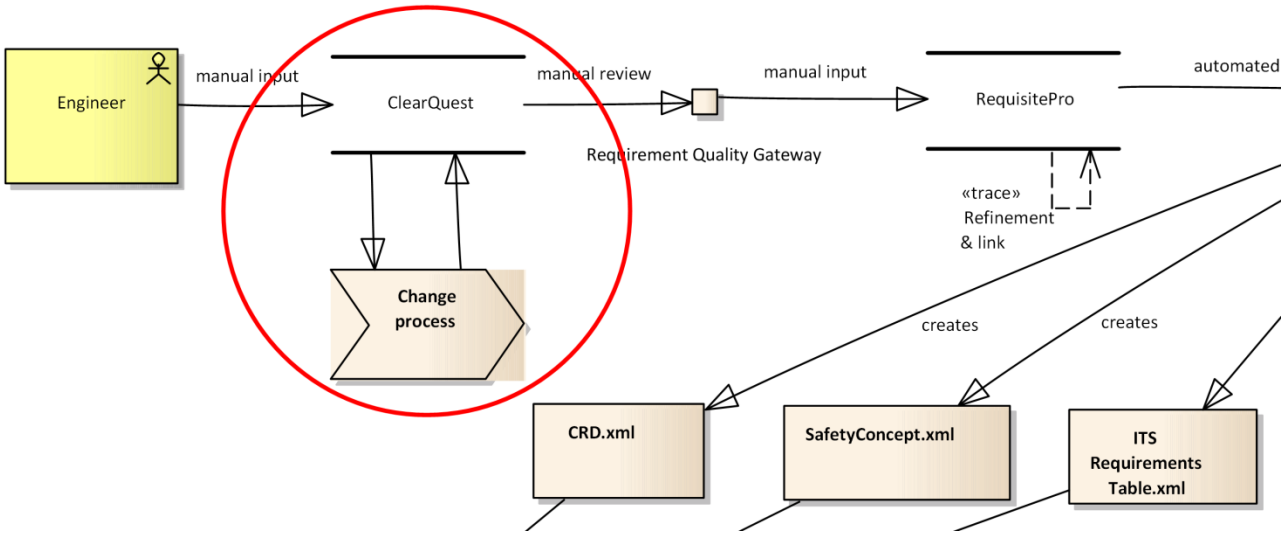


Figure 5-1: ClearQuest toolchain integration

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

6 Reqtify (B3.88)

6.1 Description

Name:	Reqtify
Contact:	Serrie.chapman@infineon.com
http://www.3ds.com/products-services/catia/capabilities/systems-engineering/requirements-engineering/reqtify	
Operating System	Windows terminal server for licenses local install (also available on terminal server)
Version	N/A
Type of Input Data	ARQE.xml
Type of Output Data	PDF Excel
Dependencies	All documentation and asuresign B3.91
License	Multisite permanent (maintenance only, limited licenses)
Additional information	Reqtify in itself is a linkage tool – we don't believe that it will require any updates but a 'type' may be needed to ensure it can read the IOS

Reqtify offers a comprehensive list of interfaces to multiple systems engineering tools. Reqtify can capture data from any source (file, database) of any vendor in a wide variety of data and file formats. It is an open and extendible platform and has interfaces to more than 60 common systems engineering tools. It can also produce Documentation from the sources, analyzing the coverage figures, which can serve as proof of requirements traceability and coverage for audit proving.

6.2 Use Case coverage and application

6.2.1 Use Case 3.3

Changes at any level may have an effect on multiple levels of implementation within the requirements tracing flow. Linking the matching requirements correctly and ensuring that refinement of the requirements is traced although it does have some issues with ensuring correctness, needing review and ensuring the tool chain is complete.

6.2.2 Requirements fulfilled by initial tool/method version

The Requirements Traceability Tree

The Requirements Traceability tree is essentially a tree of interlinked documents. Each level of the tree hierarchy can be assigned to a different part of the requirements process and it is subjective about the level of granularity and also how and where the argument or proof of correctness is.

Within Infineon we categorise into the following four sections:

- Requirements;
- Intention to implement the requirement;
- Intention to prove implementation of the requirement
- Proof of correct implementation of the requirement.

6.2.3 What will be implemented/provided in the CRYSTAL project

Reqtify will continue to link the documentation – it will be extended to ensure that it can translate the exported AREQ.xml information into the flow to be included within the Requirements Traceability Reports (RTR's)

6.3 General Improvement

6.3.1 Implementation

A first link to an example export from asuressign has been implemented. It is likely that this will need some changes throughout the lifetime of the CRYSTAL project.

The definition of Technical Items, which represent extensions, enhancements and improvements of Bricks developed in CRYSTAL to meet the UC needs, except from Interoperability, will be part of the next phase of the project.

6.4 Integration and Interoperability

6.4.1 How will this brick be integrated in the UC

Reqtify essentially acts as the glue within the documentation tree and produces audit proof documentation to show the coverage of the requirements. It is also able to analyse and extract metadata relating to the configuration management version etc so that the reports can be replicated at any time.

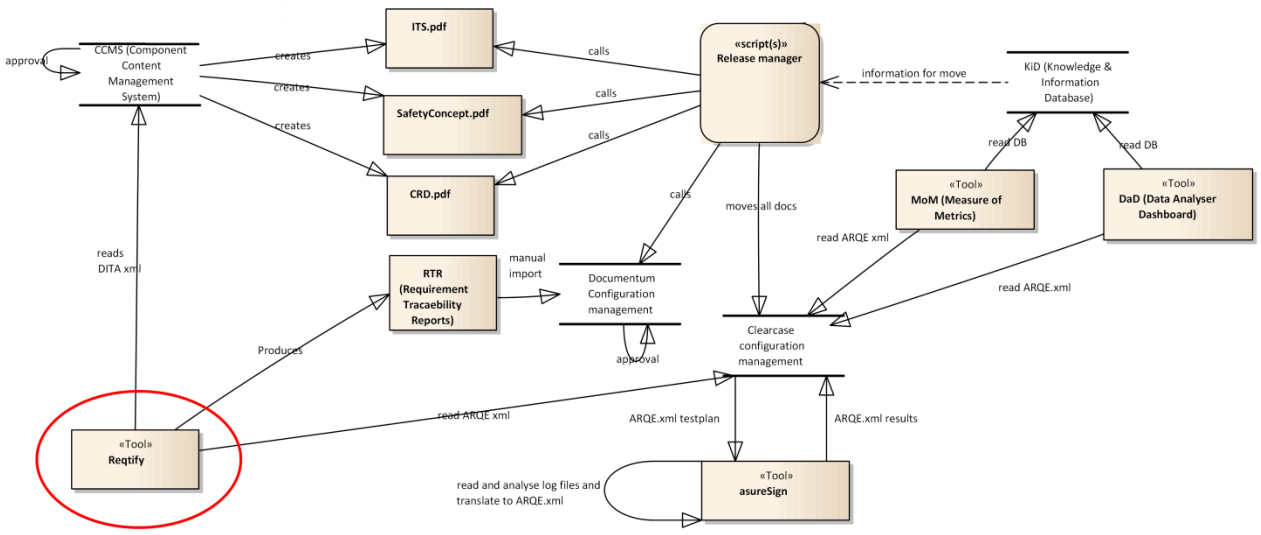


Figure 6-1: Reqtify toolchain integration

The definition of Technical Items, which represent solutions related to the interoperability of Bricks, will be part of the next phase of the project.

7 ReqIF (B3.89)

7.1 Description

Name:	ReqIF
Contact:	Serrie.chapman@infineon.com
http://en.wikipedia.org/wiki/Requirements_Interchange_Format	
Operating System	Any
Version	N/A
Type of Input Data	Xml schema
Type of Output Data	Xml schema
Dependencies	requsitePro/Clearquest B8.6 B 8.7 (possibly the CESAR DODT tool)
License	None – free download
Additional information	We will be looking at any possible harmonisation of ReqIf and ARQE.xml

RIF/ReqIF (Requirements Interchange Format) is an XML file format that can be used to exchange requirements, along with its associated metadata, between software tools from different vendors. The requirements exchange format also defines a workflow for transmitting the status of requirements between partners. Although developed in the automotive industry, ReqIF is suitable for lossless exchange of requirements in any industry.

7.2 Use Case coverage and application

7.2.1 Use Case 3.3

7.2.1.1 What format they are in

All stakeholders agree on interface format Excel/Word/UML etc

- Agreement on pictorial or ontological meanings to remove ambiguity
- Agreement on any standard interface formats and usage
- Agreement on hierarchical requirements database structure

7.2.1.2 Requirements fulfilled by initial tool/method version

None – ARQE.xml addresses asuresign, Requisite Pro and Reqtify currently, whereas ReqIF is believed to work between RM tools only.

7.2.1.3 What will be implemented/provided in the CRYSTAL project

Ensuring that either we can extend or merge with the ReqIF to allow it to fully support further than just the RM tools. It may take the place of an extension, but we would like to ensure there is no conflict between ARQE.xml and ReqIF.

7.3 General Improvement

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

7.4 Integration and Interoperability

7.4.1 Interoperability requirements

Extension of ReqIf via an extension to ensure interoperability with an extended toolset and not just the RM tools and by extensions just requirements data – extend towards Configuration management and change management tools as well as proof information.

7.4.2 How will this brick be integrated in the UC

Currently Infineon use its own xml schema to share information across its tooling. This is the area that few will be ensuring that it can either contain a superset of the ARQE.xml or does not clash. Therefore any area within the dataflow that interacts via ARAE.xml may require integration with this standard.

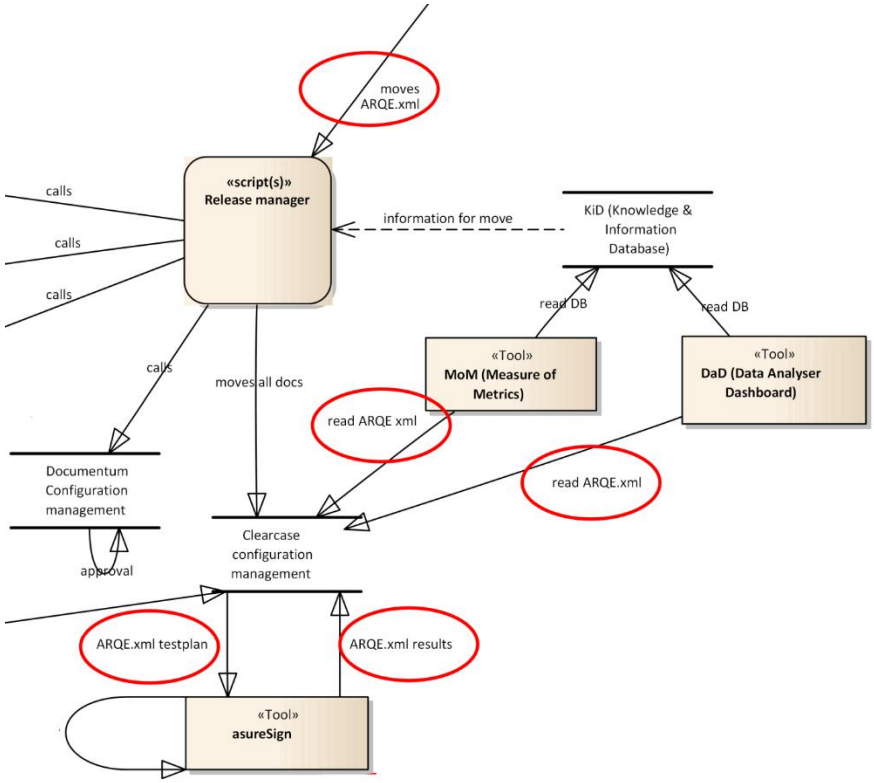


Figure 7-1: ReqIF toolchain integration

8 Documentum (B3.90)

8.1 Description

Name:	Documentum
Contact:	Serrie.chapman@infineon.com
http://www.emc.com/domains/documentum/index.htm?id=902	
Operating System	Windows
Version0	N/A
Type of Input Data	Documentation
Type of Output Data	None – storage only
Dependencies	Reqtify B3.88 release manager (internal tool)
License	Global
Additional information	

Documentum is an enterprise content management platform. Enterprise Content Management (ECM) is a formalized means of organizing and storing an organization's documents, and other content, that relate to the organization's processes. The term encompasses strategies, methods, and tools used throughout the lifecycle of the content.

8.2 Use Case coverage and application

8.2.1 Use Case 3.3

Ensure information on reviewers/comments/date etc is all saved with all of the documents produced using the requirements database as their source within a configuration management tool

8.2.2 Requirements fulfilled by initial tool/method version

Currently this is the official release storage database for all of the documents required within the audit flow so it is allowing audit proofing for the ISO26262 standard.

8.2.3 What will be implemented/provided in the CRYSTAL project

The hierarchy within Documentum will be aligned over time with the other storage area hierarchies.

8.3 General Improvement

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

8.4 Integration and Interoperability

8.4.1 Interoperability requirements

Documentum needs to have interoperability with KiD and with the release manager. This may be possible with OSLC.

8.4.2 How will this brick be integrated in the UC

Documentum has an approval and acceptance flow that is essential for ensuring good audit proof documentation storage. Although there is a new dita based flow with its own content management system integrated, this is only storage for xml formats. There are a variety of other documents such as the Requirement Traceability Reports (RTR's) and FMEDA etc which are not of an xml types and as such Documentum is required to ensure that these are safely signed off.

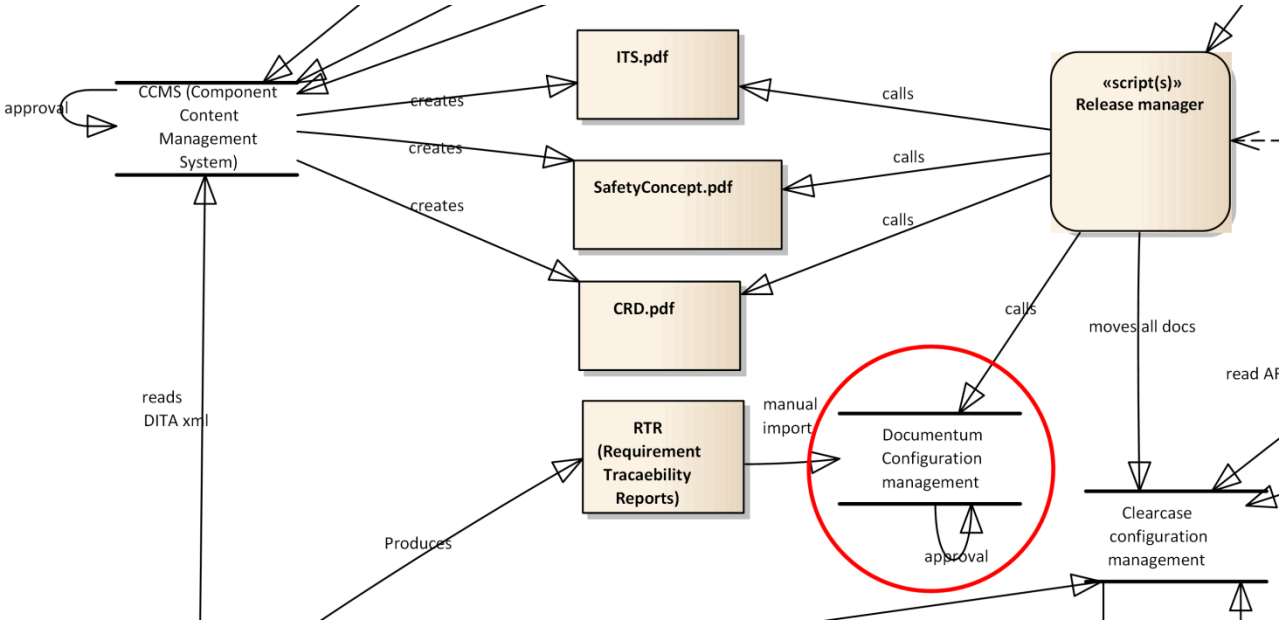


Figure 8-1: Documentum toolchain integration

9 Asuresign (B3.91)

9.1 Description

Name:	Asuresign
Contact:	Serrie.chapman@infineon.com
http://testandverification.com/solutions/requirements/	
Operating System	Windows/linux/solaris
Version	N/A
Type of Input Data	ARQE.xml
Type of Output Data	ARQE.xml/PDF
Dependencies	Requirements in ARQE.xml format
License	Open global
Additional information	This product is new and has been driven by Infineon UK to bridge the gap between Requirements and Verification results in what we term "Requirements driven verification"

Asuresign is essentially a data analyser and addresses the ‘proof of implementation’ issue. It is the bridge between the requirements Traceability flow and the Test/verification/validation proof log files. It also analyses results from reqressions overtime to assist with debug and managements of Verfication/Validation/Test projects.

9.2 Use Case coverage and application

9.2.1 Use Case 3.3

Once a requirements traceability tree has been defined then it is a question of how we can prove implementation. The ISO 26262 requires an argumented proof of a requirement being implemented correctly.

Dependant on the requirement this may be proven in many ways, examples such as ‘There shall be an FMEDA document” may be covered with the existence of the document, however most functional requirements will need tracing down to some tests reports or results. When the proving of a requirement may occur at multiple domain levels then things become more complex as the interoperability of multiple tools and results and matching those into the requirements tracing tree is not a simple task.

Within Crystal Infineon plan to deploy a new external tool called asuresign from TVS – work to allow it to link into the requirements tracing tree is currently under test. The tool essentially extracts data from regressions, which it can analyse and do comparisons from. It can also determine whether or not a requirement has been checked, define the pass criteria and translate that data for the requirements traceability reports information.

9.2.2 Requirements fulfilled by initial tool/method version

“Proof of implementation” will be fulfilled by the initial tooling. It will give the traceability flow the bility to analyse and link into the bottom of the requirements traceability flow.

9.2.3 What will be implemented/provided in the CRYSTAL project

- Full seamless integration of the tool
- Automation of the analysis of the results
- Automation of the links and results back into the requirements tracing flow
- Extension across all the domains .. pre silicon, post silicon and sw

9.3 General Improvement

9.3.1 Implementation

The Tool is currently at the end of alpha testing and is in its first rollout phase. The flow around it is being implemented within the CRYSTAL project and is at stage one of three towards completion.

The definition of Technical Items, which represent extensions, enhancements and improvements of Bricks developed in CRYSTAL to meet the UC needs, except from Interoperability, will be part of the next phase of the project.

9.4 Integration and Interoperability

9.4.1 Interoperability requirements

For interoperability asuresign fully supports the IFX ARQE.xml schema for import & export mechanisms. It is currently under analysis how to automate the moving of xml's from and to asuresign within the flow via the release manager.

9.4.2 How will this brick be integrated in the UC

Asuresign is integrated at the bottom of the toolchain as it is designed to link to the bottom of the requirement tracing tree. As there will be multiple databases across the variants and within the work products themselves relating to which test domain and which IP, Subsystem or System. It is therefore necessary to ensure the correct data is implemented for each of the databases and as such there are a variety of scripts being written and automated processes that are required to drive data in and out of the tool and back to the Requirements trace flow

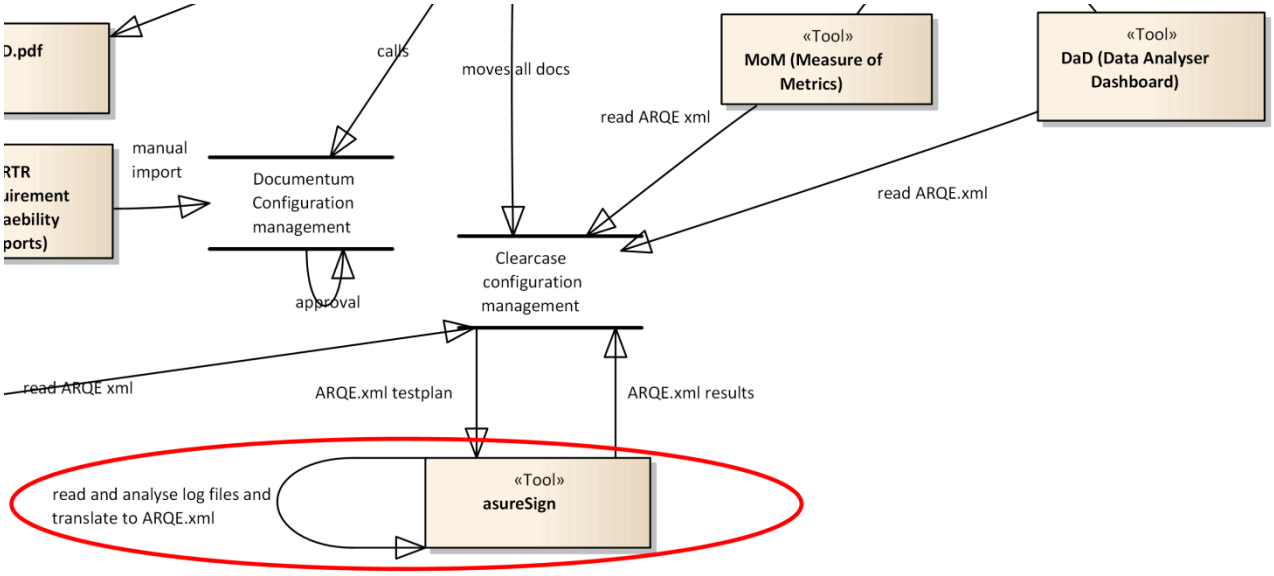


Figure 9-1: Asuresign toolchain integration

10 Rail Model (B5.1)

10.1 Description

Name:	Rail Model
Contact:	BARBERIO Gregorio (g.barberio@mateconsulting.it)
Operation System/platform	Windows
Version	N/A
Type of Input Data	Behavioural description of the system under analysis, Formal Requirements
Type of Output Data	Test cases in a formal language
Dependencies	
License	
Additional information	

This chapter discusses the specification of the B5.1 Rail Model brick – developed by Mate. Rail Model is a complete modelling environment which can be adopted to model a complex and safety-critical system (typical of the rail domain). In particular the brick shall allow the two following main functionalities:

a) the behavioural modelling of complex and asynchronous systems and the formal representation of their requirements;
b) the semi-automatic generation of functional test scripts from the model itself.

This brick can be used in a chain with other bricks of the rail domain in order to support the validation activities of a specific signalling system. In fact the output of this brick represent the input of the brick B5.3 which takes charge of transforming the test cases, here generated, in corresponding test cases written in the IOP language (a standard language for the rail domain). The outputs of the test executions are then analysed by the brick B5.4 in order to support their outcomes analysis.

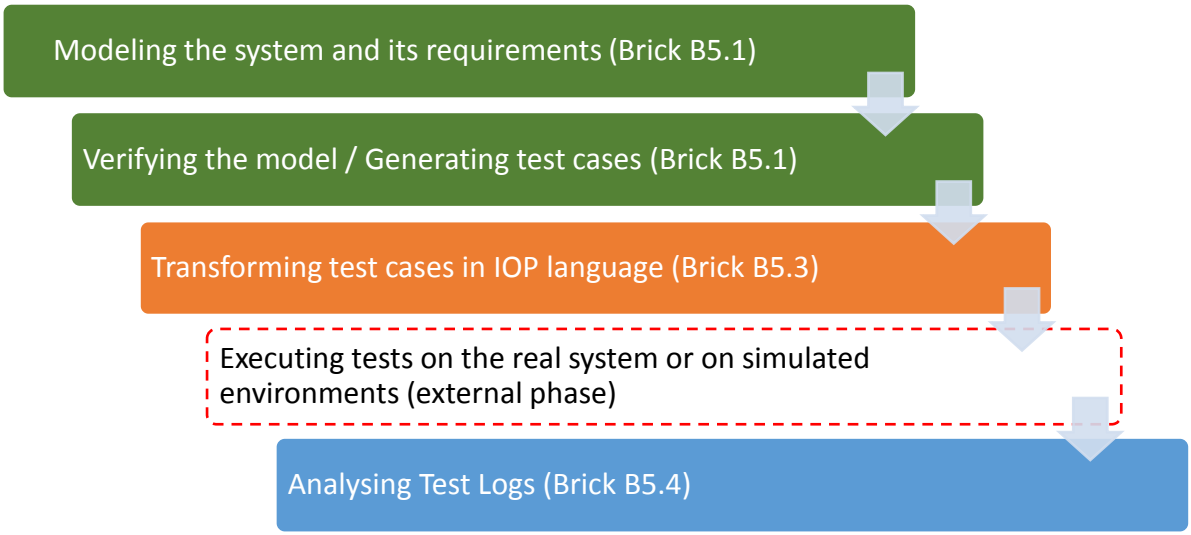


Figure 10-1: Validation workflow and supporting bricks

The figure above lists the main validation activities of a system and shows the supporting brick for each step. The step related to the execution of test case is not supported by bricks since it is performed by each railway industry with proprietary artefacts and tools.

10.1.1 Modelling the behaviour and requirements of complex and safety critical systems

The system modelling is the first step of the validation workflow; the brick B5.1 offers a complete modelling environment which helps the V&V Engineers during this activity. As highly recommended by standards applicable in the railway domain and as widely adopted by industries, this brick provides the modellers with an ad-hoc extendible modelling language based on the Finite State Machine (FSM) formalism. The FSM formalism allows defining the control structure of the system in terms of its states, possible inputs, and obtained actions. In fact safety-critical systems shall be completely specified for every input in every state: FSMs show how the system moves from one state to another as resulting from receipt of an input in a given state.

The development of the brick B5.1 shall integrate this new modelling language which is able to provide the modellers with all and only the concepts that she/he needs. The new language will implement a formal semantics and will have a clear syntax since no ambiguities are admitted during the modelling of a safety-critical system. The same modelling language shall support also the definition of Test Specifications which reflects the system Requirements listed in apposite documents.

The definition of this new modelling language is also necessary to enable the automatic generation of Test Cases [MOGENTES, 2008] with the information contained into source models. The generated Test Cases are also expressed with the same modelling language in order to be opened, read and edited with the same brick B5.1.

Hence the brick shall implements the following functionality for what concerning system modelling:

- definition of a the system model in terms of state machines;
- definition of communication mechanisms between state machines;
- modelling of test specifications and annotating of system related system requirements;
- static analysis of the model;
- generation of simulation traces from the source model.

10.1.2 Semi-automatic test case generation

After the complete definition of the system model and of the test specifications, it shall be possible to generate test cases [Javed, 2007]. The brick B5.1 supports this activity allowing their automatic generation.

The brick shall allow the traceability between test specifications and test cases. The adopted technique which helps the test case generation is based on model checking [Gargantini, 1999], , that allow to not generating test cases only if the test specification is effectively infeasible; otherwise the brick eventually generates the test case.

Generated test cases are transformed back into the source formalism in order to be showed to V&V engineers with the same graphical editor and the same language used to model the system. Generated Test Cases can be also edited by V&V Engineer, adding their specific knowledge of the system and of the domain.

10.2 Use Case coverage and application

To achieve the objectives described in the previous paragraph, below a brief introduction to one of use cases of the rail domain deeply described the deliverable D501.020.

The Radio Block Centre (RBC) is the basic component of the ETCS level2 signalling architecture: it is placed in a special central place from which the system, via Command and Control, is kept under control; it also manages the movement of trains. By means of radio transmission based on GSM-R standard, each train receives all the parameters to be respected as:

- speed;
- constraints relative to the path;
- possible temporary slowdowns;
- text messages or other information;

It is necessary to maintain the proper spacing between trains in circulation. All trains report automatically their exact position and direction of travel to the Radio Block Centre (RBC), at regular intervals. The Radio Block Centre monitors train movements continually. The movement authority is transmitted to the vehicle continuously via GSM-R together with speed information and route data. The information coming from the Radio Block Centre is elaborated and displayed on the locomotive dashboard (DMI), which will indicate the target rate, the free distance ahead and a range of information including the maximum allowed speed.

Each new implementation of the RBC requires the application of a new validation cycle which requires the definition of test cases, their execution on the system and the log analysis. In particular the generation of test cases is actually performed manually with a great effort in terms of time and costs. This effort corresponds to a great investment in terms of involved people and to, sometimes, delayed deadlines (with the payment of economic penalties in some cases). The brick B5.1 shall help the V&V engineers in performing this activity, reducing time and costs of the test case generation step.

Another important improvement given by the adoption of this brick resides in the supporting of regression testing [Jouault, 2006]: this brick and the others mentioned above (B5.3 and B5.4) shall help the V&V Engineers to invalidate a set of test cases after an update of the source model (those invalid given the update) and to regenerate them in a revised version. In this way it would be also possible to maintain artefacts between similar projects.

10.3 General Improvement

Here a description of the Technical Items offered by this Brick.

TI NAME: Non-functional improvements					
TI_ID	CRYSTAL_TI_612_10_1	**Kind of TI**	G	**Contact email**	TBD
Description: In this technical item are achieved the following results: • Storage: The system contains a unique repository for the storage of all data managed by tools belonging to the chain. The requirement can be satisfied having the possibility to navigate through the contents of different archive using a common interface. • Security: An authentication process is required for accessing the functionality of the system. • Scalability/Expandability: The system should be able to handle the increasing size of the data managed. • Usability: The system must be easy to use. A user interface should give access to all system functionalities providing easy navigation through all features.					
Link to internal working documents:					

TI NAME: Support in the creation of System Models					
TI_ID	CRYSTAL_TI_612_10_2	**Kind of TI**	G	**Contact email**	TBD
Description: This technical item consists of modelling the behaviour and requirements of complex and asynchronous systems, taking into account composition of different subparts. The modelling language adopted in the tool is an ad-hoc language, defined for railway systems. The system supports the creation, editing and updating of the system model.					
Link to internal working documents:					

TI NAME: Generation of System Tests					
TI_ID	CRYSTAL_TI_612_10_3	**Kind of TI**	G	**Contact email**	TBD
Description: This technical item is a solution to achieve two objectives: definition of test specification from the model itself and semi-automatic generation of test cases. A Test Specification verifies one or more requirements: this link shall be clearly traced; in this way, after the test execution phase, it's possible to track requirements not correctly implemented and test cases failed. Within the Generation of test cases, Test Specifications are translated in a formal language comprehensible by the V&V Engineers; in this way it's possible to operate on Test Cases and to have an idea of the model portion stressed by each Test Case.					
Link to internal working documents:					

10.4 Integration and Interoperability

Here a description of solution based on Technical Items allocated for this Brick.

TI NAME: Shared data consistency and Traceability					
TI_ID	CRYSTAL_TI_612_10_6	**Kind of TI**	I	**Contact email**	TBD
Description:					

The results achieved in this technical item are divided in two parts: Shared data consistency and Traceability. Below are described in more detail:

Shared data consistency : Some data is shared between the entire tool chain, for example:

- model that is created in the first tool, is used for the generation of test cases in the subsequent steps of the tool chain.

For these reasons, the system must ensure that the data shared within and outside the tool chain have the specification of consistency. If this requirement is not satisfied all results are ambiguous.

The consistency of the data shared between the various tools is achieved by means of a software layer that implements the collaborative environment between the different tools and defines the ways in which the tools are able to access this data. It is important to specify that between the various tools there isn't an exchange of information but a data sharing for collaborative purposes. The tools have direct access to shared data, then if the environment of sharing does not ensure a consistent representation of the data, the results of the entire tool chain may not be reliable.

Traceability between the system requirements and test cases: The system must ensure traceability between the failed tests and requirements that are not correctly implemented due to the failure. This requirement is very important because it allows identifying, in the analysis phase, the causes of the failure of the test case; otherwise, the end-user does not find the condition to be corrected. In addition, the system must ensure that in a not failed test case, there are not requirements not correctly implemented. Obviously, the traceability requirement is not satisfied if the collaborative environment does not guarantee the specification of data consistency. Then this requirement is based on the consistency requirement. This is possible through linking of different objects (i.e. Requirements and safety artifacts shall be linked to test models and test results for test coverage analysis);

Link to internal working documents:	

TI NAME: Semi-automatic modification of test cases					
TI_ID	CRYSTAL_TI_612_10_7	**Kind of TI**	I	**Contact email**	TBD

Description:

This technical item is a solution to the semi-automatic modification of tests case after a modification of system requirements. Here some example about the possible behaviours supported by the tool chain:

- if the user change the requirement of a system, the system shows the tests related to the requirement that has been changed;
- if a requirement is deleted from the model, the system automatically removes the test that involves only the requirement eliminated
- if a new requirement is added into the model, automatically, it must also be generated tests for the new requirement.
- Warning when there are newer versions of the inputs used;

Link to internal working documents:	

TI NAME: Collaborative environment resource management					
TI_ID	CRYSTAL_TI_612_10_8	**Kind of TI**	I	**Contact email**	TBD

Description:

This technical item is a set of results for collaborative environment resource management. Below are described in more detail:

- Versioning of files / managing of history;
- Commit atomicity (it should be NOT possible to have partial file committed);

• Obligatory description (each time a file is saved/modified/deleted, the user shall provide a rationale of the action); • Mutual exclusion of single file (smart management of contemporary modifications to the same file); • Warning when used inputs are locked by other users (more tools can access to the same file; if a tool/user is accessing as read-only to a file, it shall be warned whether the same file is under modification by another tool/user); • Warning for the insertion of new files in project; • File can belong to multiple projects (management of tags or similar methodologies to allow a single file to belong to different projects/workspaces).	
Link to internal working documents:	

11 IOP test writer (B5.3)

11.1 Description

Name:	IOP test writer
Contact:	BARBERIO Gregorio (g.barberio@mateconsulting.it)
Operation System/platform	Windows
Version	N/A
Type of Input Data	Test cases in a formal language
Type of Output Data	Test cases in a standard language
Dependencies	Rail Model
License	
Additional information	

IOP test writer consists of the implementation of a tool that allows the creation of test scripts defined in IOP language. The tool should be used in a chain as described in §10. The test scripts are generated by Rail Model in a meta-language that allows the tool to be used also on different industrial domains.

The test case meta language, depends strictly on the languages and formalities adopted in the tools described in the brick B5.1, moreover the file format depends directly on the formats supported by the tools used. Now the test cases must be converted into a language specific for rail domain independent from the tools previously used. In this way it will be possible, in the future, to change for example the model checker without changing the tools that use the tests furthermore using a specific domain language, guarantees the technical interoperability of tests with other platform. The target IOP language, defined by UNISIG consortium (being standardizing) allow the execution of interoperable tests in a multi-suppliers environment.

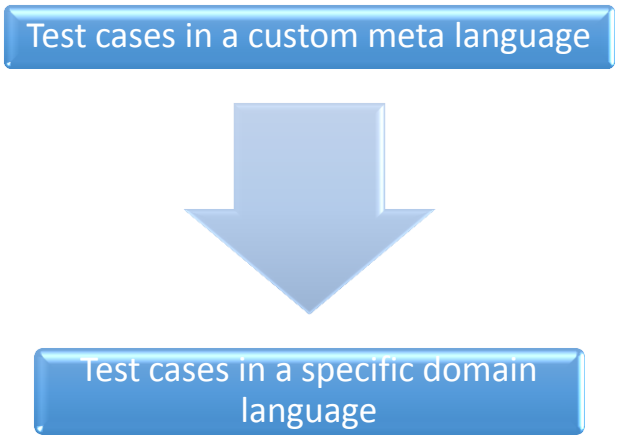


Figure 11-1: IOP Test writer tool Input-Output

11.2 Use Case coverage and application

Before introducing the requirements of the tool, here a brief introduction to the scenario.

The European Rail Traffic Management System (ERTMS) is an initiative backed by the European Union to enhance cross-border interoperability and the procurement of signalling equipment by creating a single Europe-wide standard for train control and command systems.

Companies developing ERTMS systems includes UNISIG (Union industry of signalling) members.

The ERTMS specifications give some freedom to implement functionalities. Given this freedom, there is no 100% evidence whether a track side and on-board "fit" together even when both are compliant with the specifications.

However, a good starting point will be the use of a common language to use in the test environment; the UNISIG is responsible to define that language: IOP language.

IOP (Interoperability testing) is intended to provide a common understanding of how to organize tests for ETCS projects which helps to improve collaboration of suppliers, customers and authorities for ETCS Tests. The purpose of the standardization is to make IOP tests interoperable, that means to support cooperative tests between different suppliers.

Basically, the main requirement of the tool is the transformation of test cases, generated in a meta language, in IOP language.

After test cases generation phase ended, end-user utilizes the transform function to translate the test case in IOP language.

11.3 General Improvement

Here a description of solution based on Technical Items allocated for this Brick.

TI NAME: Non-functional improvements					
TI_ID	CRYSTAL_TI_612_10_1	**Kind of TI**	G	**Contact email**	TBD
Description: In this technical item are achieved the following results: • Storage: The system contains a unique repository for the storage of all data managed by tools belonging to the chain. The requirement can be satisfied having the possibility to navigate through the contents of different archive using a common interface. • Security: An authentication process is required for accessing the functionality of the system. • Scalability/Expandability: The system should be able to handle the increasing size of the data managed. • Usability: The system must be easy to use. A user interface should give access to all system functionalities providing easy navigation through all features.					
Link to internal working documents:					

TI NAME: Generation of System Tests					
TI_ID	CRYSTAL_TI_612_10_3	**Kind of TI**	G	**Contact email**	TBD
Description: This technical item is a solution to achieve two objectives: definition of test specification from the model itself and semi-automatic generation of test cases. A Test Specification verifies one or more requirements: this link shall be clearly traced; in this way, after the test execution phase, it's possible to track requirements not correctly implemented and test cases failed. Within the Generation of test cases, Test Specifications are translated in a formal language comprehensible by the V&V Engineers; in this way it's possible to operate on Test Cases and to have an idea of the model portion stressed by each Test Case.					
Link to internal working documents:					

11.4 Integration and Interoperability

Here a description of solution based on Technical Items allocated for this Brick.

TI NAME: Shared data consistency and Traceability					
TI_ID	CRYSTAL_TI_612_10_6	**Kind of TI**	I	**Contact email**	TBD
Description: The results achieved in this technical item are divided in two parts: Shared data consistency and Traceability. Below are described in more detail: Shared data consistency : Some data is shared between the entire tool chain, for example: • model that is created in the first tool, is used for the generation of test cases in the subsequent					

steps of the tool chain.

For these reasons, the system must ensure that the data shared within and outside the tool chain have the specification of consistency. If this requirement is not satisfied all results are ambiguous.

The consistency of the data shared between the various tools is achieved by means of a software layer that implements the collaborative environment between the different tools and defines the ways in which the tools are able to access this data. It is important to specify that between the various tools there isn't an exchange of information but a data sharing for collaborative purposes. The tools have direct access to shared data, then if the environment of sharing does not ensure a consistent representation of the data, the results of the entire tool chain may not be reliable.

Traceability between the system requirements and test cases: The system must ensure traceability between the failed tests and requirements that are not correctly implemented due to the failure. This requirement is very important because it allows identifying, in the analysis phase, the causes of the failure of the test case; otherwise, the end-user does not find the condition to be corrected. In addition, the system must ensure that in a not failed test case, there are not requirements not correctly implemented. Obviously, the traceability requirement is not satisfied if the collaborative environment does not guarantee the specification of data consistency. Then this requirement is based on the consistency requirement. This is possible through linking of different objects (i.e. Requirements and safety artifacts shall be linked to test models and test results for test coverage analysis);

Link to internal working documents:	

TI NAME: Semi-automatic modification of test cases					
TI_ID	CRYSTAL_TI_612_10_7	**Kind of TI**	I	**Contact email**	TBD

Description:

This technical item is a solution to the semi-automatic modification of tests case after a modification of system requirements. Here some example about the possible behaviours supported by the tool chain:

- if the user change the requirement of a system, the system shows the tests related to the requirement that has been changed;
- if a requirement is deleted from the model, the system automatically removes the test that involves only the requirement eliminated
- if a new requirement is added into the model, automatically, it must also be generated tests for the new requirement.
- Warning when there are newer versions of the inputs used;

Link to internal working documents:	

TI NAME: Collaborative environment resource management					
TI_ID	CRYSTAL_TI_612_10_8	**Kind of TI**	I	**Contact email**	TBD

Description:

This technical item is a set of results for collaborative environment resource management. Below are described in more detail:

- Versioning of files / managing of history;
- Commit atomicity (it should be NOT possible to have partial file committed);
- Obligatory description (each time a file is saved/modified/deleted, the user shall provide a rationale of the action);

• Mutual exclusion of single file (smart management of contemporary modifications to the same file); • Warning when used inputs are locked by other users (more tools can access to the same file; if a tool/user is accessing as read-only to a file, it shall be warned whether the same file is under modification by another tool/user); • Warning for the insertion of new files in project; • File can belong to multiple projects (management of tags or similar methodologies to allow a single file to belong to different projects/workspaces).

Link to internal working documents:	

12 Log Analyzer (B5.4)

12.1 Description

Name:	Log Analyzer
Contact:	BARBERIO Gregorio (g.barberio@mateconsulting.it)
Operation System/platform	Windows
Version	N/A
Type of Input Data	Execution log in a custom format
Type of Output Data	Report in pdf format
Dependencies	Rail Model / IOP test writer
License	
Additional information	

The Log Analyzer tool represents the implementation of a tool that relying on the tracing information written by RailModel, allows the easy identification of failed tests and, more important, of the requirements non-right implemented. The result of the test campaign is written in a test report document. This tool should be used in a chain as described in §10.

End-user selects this function by interacting with the GUI of tool chain and have access to log analyzer tool. To continue with the analysis phase, the tests must have been performed (the test execution task is not part of the tool chain) and the tests results must be available (we suppose the test execution phase produce a log in a known format).

Through this feature, it is possible:

- to identify failed tests;
- to analyse failed tests;
- to find the requirements not correctly implemented.

12.1.1 Failed test identification

In this step, end-user identifies the failed tests. To access failed tests, the user, after loading the appropriate log, starts the process of displaying failed tests. The tool responds to the user request by showing the set of tests in which there has been a failure.

Test ID	Test status	Requirements passed	Requirements not passed
TS_1	OK	Req1, Req2	
TS_2	OK	Req2, Req3, Req4	
TS_3	**KO**	Req2, Req3, Req5	**Req4**

Table 12-1: Example of report for failed tests

For every test, this function shows:

- Test ID, a unique identifier of a test;
- Test status, OK - if the test case is not failed / KO - if the test case is failed;
- Requirement passed, requirements satisfied by OK tests;
- Requirement not passed, requirement not satisfied by tests;

12.1.2 Requirements not correctly implemented identification

An alternative report produced by the tool shows the requirements not satisfied, for each requirement the following information are displayed:

- Req ID, a unique identifier of the requirement, valid in any point of chain;
- Req Status, OK – if all tests that have impact on the requirement are satisfied / KO – if at least a test that have impact on the requirement has failed;
- Test passed, the lists of the tests correctly executed;
- Test failed, the lists of the tests failed.

Req ID	Req status	Tests passed	Tests not passed
Req1	OK	TS_1	
Req2	OK	TS_1, TS_2, TS_3	
Req4	**KO**	TS_1, TS_2	**TS_3**

Table 12-2: Example of report for failed Requirements

To identify the requirements not correctly implemented that caused failed test, the tool implements a selection function that allow the user to visualize the execution of the state machine that generated the failure of the system.

12.2 Use Case coverage and application

12.2.1 [B5.4] LOGANALYZER TOOL

The last tool of the chain explained in Brick 5.1 support the users in the analysis of test results. The used method is called the oracle, and is often applied by an automated agent to generate the correct results to be used as comparison during the test. In this phase an execution log, generated by a system external to the chain, and the test cases (described in a meta language), generated by the previous tool, are given as input to the analyzer that will produce a set of reports about the analysis.

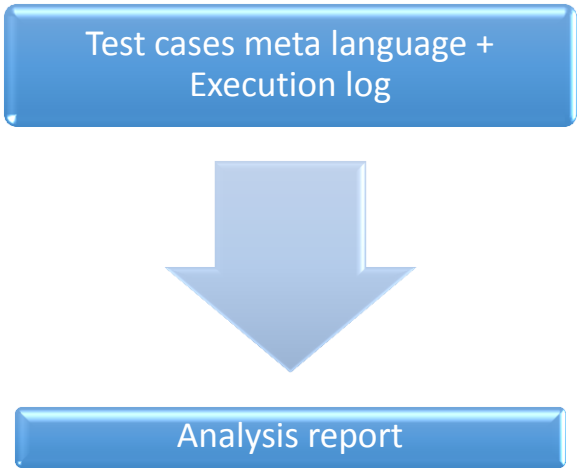


Figure 12-1: LogAnalyzer Tool Input-Output

The oracle and the model to be analyzed must share identical test conditions, input data and test environment.

12.3 General Improvement

Here a description of solution based on Technical Items allocated for this Brick.

TI NAME: Non-functional improvements					
TI_ID	CRYSTAL_TI_612_10_1	**Kind of TI**	G	**Contact email**	TBD
Description: In this technical item are achieved the following results: • Storage: The system contains a unique repository for the storage of all data managed by tools belonging to the chain. The requirement can be satisfied having the possibility to navigate through the contents of different archive using a common interface. • Security: An authentication process is required for accessing the functionality of the system. • Scalability/Expandability: The system should be able to handle the increasing size of the data managed. • Usability: The system must be easy to use. A user interface should give access to all system functionalities providing easy navigation through all features.					
Link to internal working documents:					

TI NAME: Generate Test Report					
TI_ID	CRYSTAL_TI_612_10_4	**Kind of TI**	G	**Contact email**	TBD
Description: This technical item is a solution to generate test report after tests execution.					
Link to internal working documents:					

12.4 Integration and Interoperability

Here a description of solution based on Technical Items allocated for this Brick.

TI NAME: Shared data consistency and Traceability					
TI_ID	CRYSTAL_TI_612_10_6	**Kind of TI**	I	**Contact email**	TBD

Description:

The results achieved in this technical item are divided in two parts: Shared data consistency and Traceability. Below are described in more detail:

Shared data consistency : Some data is shared between the entire tool chain, for example:

- model that is created in the first tool, is used for the generation of test cases in the subsequent steps of the tool chain.

For these reasons, the system must ensure that the data shared within and outside the tool chain have the specification of consistency. If this requirement is not satisfied all results are ambiguous.

The consistency of the data shared between the various tools is achieved by means of a software layer that implements the collaborative environment between the different tools and defines the ways in which the tools are able to access this data. It is important to specify that between the various tools there isn't an exchange of information but a data sharing for collaborative purposes. The tools have direct access to shared data, then if the environment of sharing does not ensure a consistent representation of the data, the results of the entire tool chain may not be reliable.

Traceability between the system requirements and test cases: The system must ensure traceability between the failed tests and requirements that are not correctly implemented due to the failure. This requirement is very important because it allows identifying, in the analysis phase, the causes of the failure of the test case; otherwise, the end-user does not find the condition to be corrected. In addition, the system must ensure that in a not failed test case, there are not requirements not correctly implemented. Obviously, the traceability requirement is not satisfied if the collaborative environment does not guarantee the specification of data consistency. Then this requirement is based on the consistency requirement. This is possible through linking of different objects (i.e. Requirements and safety artifacts shall be linked to test models and test results for test coverage analysis);

Link to internal working documents:	

TI NAME: Semi-automatic modification of test cases					
TI_ID	CRYSTAL_TI_612_10_7	**Kind of TI**	I	**Contact email**	TBD

Description:

This technical item is a solution to the semi-automatic modification of tests case after a modification of system requirements. Here some example about the possible behaviours supported by the tool chain:

- if the user change the requirement of a system, the system shows the tests related to the requirement that has been changed;
- if a requirement is deleted from the model, the system automatically removes the test that involves only the requirement eliminated
- if a new requirement is added into the model, automatically, it must also be generated tests for the new requirement.
- Warning when there are newer versions of the inputs used;

Link to internal working documents:	

TI NAME: Collaborative environment resource management					
TI_ID	CRYSTAL_TI_612_10_8	**Kind of TI**	I	**Contact email**	TBD

Description:

This technical item is a set of results for collaborative environment resource management. Below are described in more detail:

- Versioning of files / managing of history;
- Commit atomicity (it should be NOT possible to have partial file committed);
- Obligatory description (each time a file is saved/modified/deleted, the user shall provide a rationale of the action);
- Mutual exclusion of single file (smart management of contemporary modifications to the same file);
- Warning when used inputs are locked by other users (more tools can access to the same file; if a tool/user is accessing as read-only to a file, it shall be warned whether the same file is under modification by another tool/user);
- Warning for the insertion of new files in project;
- File can belong to multiple projects (management of tags or similar methodologies to allow a single file to belong to different projects/workspaces).

Link to internal working documents:	

13 Embedded Verification Platform (B3.100)

13.1 Description

Name:	Integrated Tool Environment for embedded controls development
Contact:	Joerg Settelmeier (Joerg.Settelmeier@avl.com)
Operation System/platform	Windows / Matlab Simulink
Version	V2.4.2.5
Type of Input Data	Formal Requirements, Matlab Simulation models
Type of Output Data	Test Cases
Dependencies	nothing
License	N/A
Additional information	N/A

Various tools are used for the different development steps throughout the software development V-Cycle as illustrated in Figure 13-1.

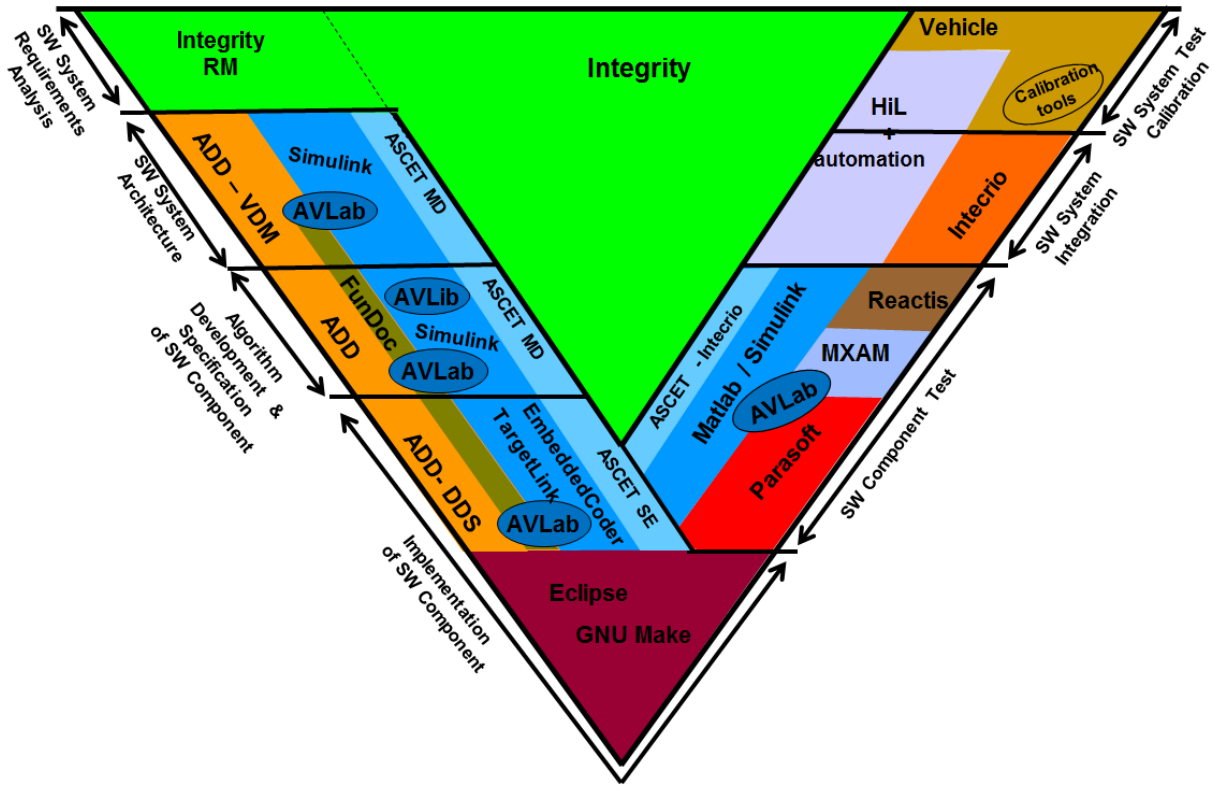


Figure 13-1: Development process including tool landscape

AVL-R is currently using the *Integrated Tool Environment* (AVLab) as a common user interface which is a single point of contact for all related tools, i.e. AVLab bundles several tools supporting each development activity steps from modeling over testing to code generation => **seamless tool chain**

- ADD (Visu-IT!) ⇔ Simulink: Data synchronization via SyncTool
- Integrity (PTC) ⇔ Matlab: Support and Integration of PTC Integrity Source in AVLab
- Simulink (The Mathworks)⇔ AVL Concerto: allowed Concerto Plots for data visualization via AVLab
- MXAM (MES) with AVL modeling rules is started from AVLab
- MIL / SIL / back2back tests supported by AVLab
- Code Generation

It also provides some kind of guidance for the developer through the development process (requirement management, architecture, model development, tests, and code generation).

Further background is the harmonization of the process & tool environment for PTE Controls.

Key points for this harmonization are:

- **Component based development** approach is enforced.
- **Scheduling** of components in function groups (and domain) is enforced by model template in AVLab.
- **SW Architecture** is based on PSF definitions and enforced by ADD as architecture tooling.
- **Code generation** for SW components is supported by tool environment in Embedded Coder and Target Link. Code generator configuration is unified.
- **Build environment** (= generation of flash able hex file) is based on generated and archived C-Code.
- Integrated **test framework** enforces common way of testing.

AVLab also shall support function development from model development over model testing to **code generation** in a Matlab/Simulink environment, with Embedded Coder or TargetLink as code generator. In addition, AVLab shall support the **methods** in engineering area:

- **A 3 Level Architecture** is the basement of the component-based development
- **Naming** is ensured by the usage of the Name Checker inside ADD
- **Modeling Guidelines** are checked by the usage of MXAM (model style checker)
- **Product Documentation** is ensured by the usage of FunDoc (Visu-IT)
- **Verification/Validation** is supported via MiL, SiL and Back-to-back testing in AVLab
- **Coding Guidelines** are supported via Code Generation Helpers (Embedded Coder Toolbar or TL Code Generator)
- **Build** is directly supporting component-based approach

13.2 Use Case coverage and application

AVLab is used only in WP3.4b.

The current implementation of AVLab increases the efficiency / quality of function development.

- Provide a template for modeling, with operating system to allow a simulation closer to the reality than a pure Simulink simulation.
- Simulink toolbar, with shortcuts for faster action in Simulink
- Traceability between Model, ADD Container, Integrity

13.3 General Improvement

The interconnection between several tools and AVLab is currently realized in a not standardized way and has to be improved.

For an improvement it has to be analyzed about an IOS implementation. Furthermore, the “Integrated Tool Environment” provides a single point of control during all development steps including MiL/SiL Tests. An extension to support automatically HiL / Engine Bench tests which are currently done manually.

13.4 Integration and Interoperability

AVLab needs to improve its interoperability capabilities with the tool chain.

The aim (*interoperability challenge)* is to provide a standardized interface for the “Integrated Tool Environment” in order to harmonize interfaces, facilitate the substitution of tools, and to be more independent of concrete tool versions. Furthermore, seamless traceability between all artifacts should be supported.

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

TI NAME: Analysis of IOS and OSLC usage					
TI_ID	CRYSTAL_TI_612_13_1	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

TI NAME: Piloting of IOS/OSLC with Integrity					
TI_ID	CRYSTAL_TI_612_13_2	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

TI NAME: Analysis of needed interfaces between AVLab and Integrity					
TI_ID	CRYSTAL_TI_612_13_3	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

14 Terms, Abbreviations and Definitions

ADD	Automotive Data Dictionary
CESAR	**C**ost-**E**fficient methods and processes for **SA**fety **R**elevant embedded systems
CRYSTAL	**CR**itical S**YST**em Engineering **A**cce**L**eration
CO	Confidential, only for members of the consortium (including the JU).
D	Demonstrator
DMI	Driver-Machine Interface
ERTMS	European Rail Traffic Management System
ETCS	European Train Control System
FSM	Finite State Machine
GSM-R	Global System for Mobile Communications – Railway
GUI	Graphical User Interface
HiL	Hardware in the Loop
IOP	Interoperability testing
IOS	Interoperability Specification
MAGIC	Measurement Analysis & GraphICs tool from AVL for measured data pre/post-processing
MiL	Model in the Loop
O	Other
OSLC	Open Services for Lifecycle Collaboration
P	Prototype
PP	Restricted to other program participants (including the JU).
PROMELA	Process/Protocol Meta Language
PSF	Powertrain Software Framework
PU	Public
R	Report
RBC	Radio Block Centre
RE	Restricted to a group specified by the consortium (including the JU).
ReqIf	Requirements Interchange Format
SiL	Software in the Loop
SP	Subproject
TSR	Temporary Speed Restriction
UES	Unconditional Emergency Stop
UML	Unified Modeling Language
UNISIG	Union industry of signalling
VEVAT	VErification & VAlidation Tool from AVL for software resp. system testing
V&V	Verification and Validation
WP	Work Package
XMI	XML Metadata Interchange

Table 14-1: Terms, Abbreviations and Definitions

15 References

CESAR	http://www.cesarproject.eu/
[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.
[MOGENTES, 2008]	MOGENTES research team, MOGENTES: Model-Based Generation of Test-Cases for Embedded Systems - State of the Art Survey - Part a: Model-based Test Case Generation Techniques Vers. 1-19a 1.1r; 2008.
[Jouault, 2006]	F. Jouault and I. Kurtev; Transforming models with ATL; In Satellite Events at the MoDELS 2005 Conference, pages 128–138. Springer, 2006.
Frank Budinsky	Eclipse Modeling Framework: A Developer's Guide
E Biermann, K Ehrig, C Köhler, G Kuhns	Graphical definition of in-place transformations in the eclipse modeling framework
DS Kolovos, LM Rose, RF Paige	Raising the level of abstraction in the development of GMF-based graphical model editors
K Ehrig, C Ermel, S Hänsgen, G Taentzer	Generation of visual editors as eclipse plug-ins

Table 15-1: References

16 Annex

No annex in this document version.