PROPRIETARY RIGHTS STATEMENT

THIS DOCUMENT CONTAINS INFORMATION, WHICH IS PROPRIETARY TO THE CRYSTAL CONSORTIUM. NEITHER THIS DOCUMENT NOR THE INFORMATION CONTAINED HEREIN SHALL BE USED, DUPLICATED OR COMMUNICATED BY ANY MEANS TO ANY THIRD PARTY, IN WHOLE OR IN PARTS, EXCEPT WITH THE PRIOR WRITTEN CONSENT OF THE CESAR CONSORTIUM THIS RESTRICTION LEGEND SHALL NOT BE ALTERED OR OBLITERATED ON OR FROM THIS DOCUMENT. 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

Brick Interfaces
D502.020

DOCUMENT INFORMATION

Project	CRYSTAL
Grant Agreement No.	ARTEMIS-2012-1-332830
Deliverable Title	Brick Interfaces
Deliverable No.	D502.020
Dissemination Level	CO
Nature	R
Document Version	V1.01
Date	2014-04-30
Contact	Rupert Schlick
Organization	AIT
Phone	+43 50550 4124
E-Mail	rupert.schlick@ait.ac.at

AUTHORS TABLE

Name	Company	E-Mail
Rupert Schlick	AIT	rupert.schlick@ait.ac.at

CHANGE HISTORY

Version	Date	Reason for Change	Pages Affected
V0.01	2014-01-14	Initial version	All
V0.02	2014-04-28	Refined structure, pre-Final Section 3.2	All
V0.03	2014-04-28	Extended section 4	(section 4)
V0.04	2014-04-28	Ready for WP internal Review	All
V1.00	2014-04-29	Reviewed by TRAIL	All
V1.01	2014-04-30	Integrated WP external review comments from TTTech	2,3,7,8,9,11,12,14, 15

CONTENT

Content of Figures

Content of Tables

1 Introduction

1.1 Role of deliverable

This document is the second deliverable of the WP5.2 of the CRYSTAL project. It describes the interactions and interfaces for the bricks to be used in the use case.

1.2 Relationship to other CRYSTAL Documents

Builds on D502.010 TRAIL Use Case Definition.

Provides foundations for D604.902 Safety Tools Specification, Development and Assessment Report V2 and D502.902 TRAIL Use-Case Development Report - V2

1.3 Structure of this document

Chapter 2 briefly summarizes the development process and artefacts as elaborated in D502.010.

Chapter 3 gives an overview of the overall brick interactions and describes the scenarios in which tools interact or interoperate.

Chapter 4, finally, defines the interfaces used, also giving an initial gap analysis, describing the existing assets and highlighting the developments done within the project.

2 Use Case Context

The main new aspects in the systems engineering environment for this use case are model based safety analysis and model based test case generation, accompanied with improved automated tracking and reporting of verification and validation activities.

Figure 2-1 shows the overview over artefacts and tools for the whole development process, based on the V-model as blocks, with process steps deriving one artefact from another as arrows. Arrows highlighted in red and green are the above mentioned new aspects being in the centre of attention for this use case.

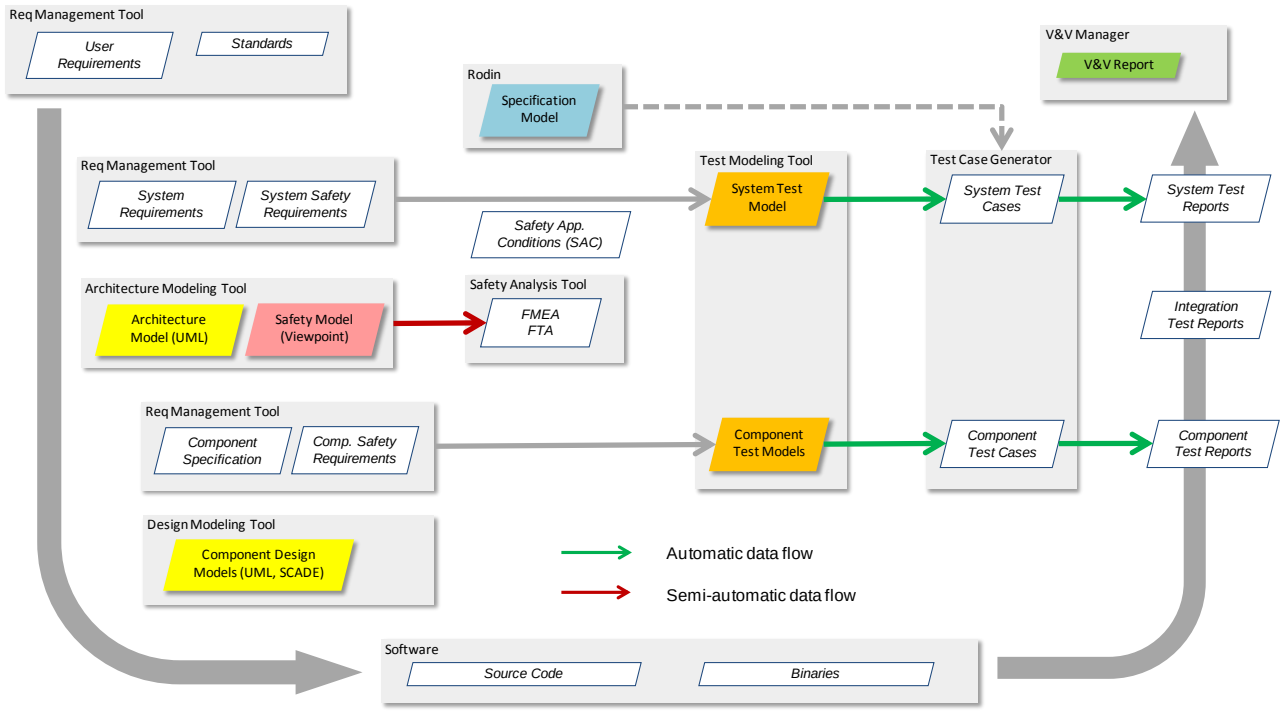

Figure 2-1: Tools and artefacts in the development process.

3 Interaction

3.1 Development Activities Overview

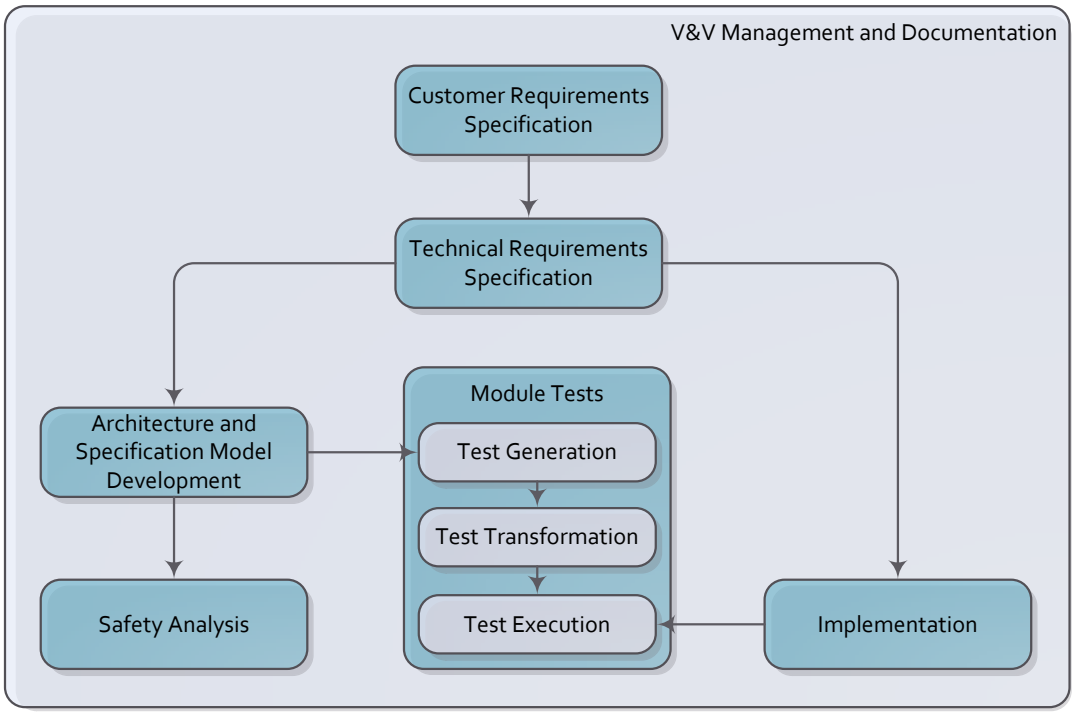

Figure 3-1: Development Activities

For defining tool interaction scenarios, we focus on the activities needed to create and maintain the artefacts given in Figure 2-1, rather than the tools used to do so. Figure 3-1 gives an overview of the development activities mainly addressed in the use case and general dependencies between them, without giving explicit artefacts and tools. In the next subsection, tool interaction scenarios for these activities are described. Source code version management is kept as is and not included in the interaction scenarios. Nonetheless, references to artefacts under version control need to provide means to document the referenced version.

3.2 Tool Interaction Scenarios

In this section, for the development tasks within the scope of the use case, activities/situations where a user will need or manipulate information from more than one tool are given in requirements style where “shall” is used to express the requirement nature of the descriptions. Non-mandatory aspects use “should”. The scenarios try to follow the expressed policy at TRAIL that a developer should be able to do as much as possible from within the Eclipse development environment.

There are four types of interactions planned:

- Establishing and following traceability links
- Import of data (structure)
- Automated execution
- Information on updates/changes of elements in another tool

3.2.1 Technical Requirements Specification

Used integration types:

- Establishing and following traceability links

When defining technical requirements link references to user requirements, system variants and scenarios shall be recorded. The requirements editor embedded in Eclipse shall provide a possibility to add a reference. This shall open a selection dialog from the respective source and establish the link on both sides.

When reviewing requirements, a preview of the related artefacts (user requirements, variants, scenarios) shall be available.

3.2.2 Architecture and Specification Model Development

Used integration types:

- Establishing and following traceability links

Specification models, further refining the technical requirements, shall trace back to the technical requirements. In order to enter these requirement references, a dialog for adding them to the model shall be provided.

When using UML models, a way to fill a stereotype tag slot with a reference to a (technical) requirement shall be provided. The stereotypes and tags to fill shall be either defined in the SysML profile or in a proprietary profile. For Event-B as specification model, establishing links from model elements to requirements shall be provided in the editor. In both cases, previews of the requirements in the editor shall be available. If possible, the link should be navigable from both sides – requirement to model element and model element to requirement.

3.2.3 Safety Analysis

Used integration types:

- Establishing and following traceability links
- Import of data (structure)

The architecture of the system shall be imported from the specification model. References back to the origin are to be kept in sync. Results of the safety analysis may lead to additional (safety) requirements and additional architecture elements – pre-filled creation dialogs, together with implicit linking to the respective safety analysis result shall be provided.

3.2.4 Implementation

Used integration types:

- Establishing and following traceability links

It shall be possible to easily embed references to requirements and to specification model elements into the source code (e.g. via drag and drop from the requirements editor). A notation to uniquely identify source code locations shall be defined. Adding references to the source code shall establish a backlink from the requirements and model elements referenced. If possible, a preview of the referenced model from within the source code editor should be provided.

3.2.5 Module Test

Used integration types:

- Establishing and following traceability links
- Automated execution

Module tests are done in the following steps:

- Generation of test cases from the model
- Transformation from test cases to test scripts

- Execution of tests

For all three steps, remote execution without user interaction shall be provided. The following traceability links shall be persisted:

- Model element to test case (possibly including parameters, like mutation operators)
- (Abstract) test case to (concrete) test script
- Test script to test result

Queries shall be provided to give information about which test cases are new or obsolete from the last model/requirements iteration and to allow prioritisation of test cases for test execution (e.g. new tests first, shortest tests first, most effective tests i.e. tests with high coverage first)

Additionally, coverage analysis steps shall be provided, establishing coverage links to model elements and to source code locations.

Automatically creating entries in the issue management system for failed test runs shall be supported.

3.2.6 V&V Management and Documentation

Used integration types:

- Establishing and following traceability links
- Automated execution
- Information on updates/changes of elements in another tool

Verification and validation activities are planned based on requirements, standards, defined processes and safety analysis results. This is done in part automatically and in part by a V&V engineer. Automatically created plans shall be automatically linked to the artefacts they are derived from. For plans created by a user, dialogs for establishing the links shall be provided. Previews of the related elements shall be shown on user request.

There are three modes of execution of the planned activities. All three are available as soon as the input dependencies for the activity are available and the output of the activities is outdated with respect of to the version of the inputs.

- Fully automated execution of the activities
- User triggered automated execution from the user's task list
- Manual execution and documentation of the results via the user's task list

Updates of artefacts on which the execution of V&V activities depends on shall be observed automatically.

When analysing execution progress, links to related artefacts shall be provided in a way that allows preview of the artefacts.

Viewing of the user's V&V management task list shall be possible in the same way as tasks in the issue management system.

4 Tools and their interfaces

In the following, the interfaces of tools to be interconnected are described. The tools are grouped by tool classes.

There are plans at TRAIL to use tools for managing variants and scenarios in the future. These tools are not part of the use case, but are supported by foreseeing additional link targets in the technical requirements.

With the exception of the safety analysis tools, all interoperability requirements can be fulfilled by implementing currently released OSLC specifications, namely:

- Architecture Management[1]
- Asset Management[2]
- Automation[3]
- Change Management[4]
- Quality Management[5]
- Requirements Management[6]

4.1 Requirements Management

4.1.1 DOORS

DOORS is a requirements management system from IBM and already used at TRAIL for management of user requirements.

Interoperability features:
OSLC RM provider:

- requirements selection (delegated UI or OSLC query)
- requirements creation (delegated UI or OSLC requirements factory)
- requirements link creation (OSLC requirements factory)
- requirements update and requirements link update (OSLC requirements CRUD interface)
- requirements preview

Delegated user interfaces included/used:

- linked model elements preview
- linked requirements preview

Eclipse integration:

will be done via DOORS web access in an embedded browser

Availability:
DOORS provides a full OSLC RM implementation

4.1.2 ProR

ProR is an Eclipse plugin for requirements management. It's currently planned to be used for managing technical requirements. ProR is already integrated with the Event-B platform Rodin, it is planned to reuse this integration and harmonize it with the other RM interoperability approaches.

Interoperability features:

[1] http://open-services.net/wiki/architecture-management/OSLC-Architecture-Management-Specification-Version-2.0/
[2] http://open-services.net/wiki/asset-management/OSLC-Asset-Management-2.0-Specification/
[3] http://open-services.net/wiki/automation/OSLC-Automation-Specification-Version-2.0/
[4] http://open-services.net/bin/view/Main/CmSpecificationV2
[5] http://open-services.net/bin/view/Main/QmSpecificationV2
[6] http://open-services.net/bin/view/Main/RmSpecificationV2

OSLC RM provider:

- requirements selection (delegated UI or OSLC query)
- requirements creation (delegated UI or OSLC requirements factory)
- requirements link creation (OSLC requirements factory)
- requirements update and requirements link update (OSLC requirements CRUD interface)
- requirements preview

OSLC RM consumer:

- requirements link creation (OSLC requirements link factory)

Delegated user interfaces included/used:

- linked model elements preview
- linked requirements preview

Eclipse integration:

ProR is an Eclipse plugin

Availability:

Provision of an OSLC RM interface is an open feature request for ProR. Currently it is unclear how and when this will be implemented.

4.2 Modelling

4.2.1 Rodin

Rodin is an Event-B modelling environment implemented on top of the Eclipse platform. It's one of the currently planned options to create and maintain specification models. Rodin features a notion of model refinement and supports refinement checking of a new refinement towards a more abstract prior refinement.

Interoperability features:

OSLC AM provider:

- architecture element selection (delegated UI or OSLC query)
- architecture element creation (delegated UI or OSLC architecture element factory)
- architecture element link creation (OSLC architecture element factory)
- architecture element update and architecture element link update (OSLC architecture element CRUD interface)
- architecture element preview

OSLC RM consumer:

- requirements link creation (OSLC requirements link factory)

Delegated user interfaces included/used:

- requirements selection dialog
- linked requirements preview
- linked test case preview

Eclipse integration:

Rodin is an Eclipse plugin

Availability:

IOS integration has to be developed. It is not yet clear how Rodin's file based paradigm will be adapted to provide web-based IOS access to the model for linking.

4.2.2 Topcased/Papyrus MDT

Topcased is an UML modelling environment implemented as an Eclipse plugin. It uses the Papyrus MDT UML modelling component, which can also be used without the rest of Topcased. It is one of the currently planned options to create and maintain specification models.

Interoperability features:

OSLC AM provider:

- architecture element selection (delegated UI or OSLC query)
- architecture element creation (delegated UI or OSLC architecture element factory)
- architecture element link creation (OSLC architecture element factory)
- architecture element update and architecture element link update (OSLC architecture element CRUD interface)
- architecture element preview

OSLC RM consumer:

- requirements link creation (OSLC requirements link factory)

Delegated user interfaces included/used:

- requirements selection dialog
- linked requirements preview
- linked test case preview

Eclipse integration:

Topcased and Papyrus/MDT are Eclipse based tools

Availability:

In the project MBAT an integration approach for Papyrus has been developed by other partners. Reuse of this integration still has to be investigated further. It is not yet clear how Papyrus/MDT's file based paradigm will be adapted to provide web-based IOS access to the model for linking.

4.3 Safety Analysis

4.3.1 Safety Architect

Safety Architect is a safety engineering tool by All4Tec.

Interoperability features:

OSLC AM consumer:

- architecture element and architecture element link CRUD interface
- architecture element query

OSLC RM consumer:

- requirement and requirement link CRUD interface

Safety resources provider

- safety resource read
- safety resource link CRUD

Delegated user interfaces included/used:

- architecture element preview
- architecture element selection
- architecture element creation
- requirement creation

Eclipse integration:

n/a

Availability:

All4Tec is developing the IOS integration for safety architect within CRYSTAL. Safety features to be exposed via the safety resource provider are not fully defined yet (faults, barriers, failure modes, ..) and are not supported by OSLC/IOS definitions up to now.

4.4 IDE

4.4.1 Eclipse for implementation

The Eclipse Java development environment is used for implementation. An approach for referencing to source code elements is already in place. Advantages and drawbacks of changing this to use IOS will be investigated. This change could be done by providing OSLC asset management access to the code – defining source code files as asset and the respective code fragments/lines as asset fragments.

Interoperability features:

OSLC asset management provider:

- asset and artefact selection (delegated UI or OSLC query)
- artefact back-link creation (OSLC artefact link factory)
- artefact back-link update (OSLC artefact link CRUD interface)

OSLC RM consumer:

- requirements link creation (OSLC requirements link factory)

OSLC AM consumer:

- architecture element link creation (OSLC architecture link factory)

Delegated user interfaces included/used:

- requirements selection dialog
- architecture element selection dialog
- linked requirements preview
- linked architecture element preview

Eclipse integration:

Native Eclipse

Availability:

The integration needs to be developed. The achievable level of integration is currently unknown.

4.5 Test Case Generators

4.5.1 MoMuT::UML

MoMuT::UML generates test cases from UML models. Since test case generation can take substantial time and computation power, it is done in batch mode. For this use case, MoMuT::UML will also act as OSLC quality management system. In case there is another OSLC QM provider, QM related resources can also be stored there. Test case transformation to test scripts will be possible via plugins, which can also be scheduled as automation tasks. Pre-existing tests can be imported via the OSLC QM provider interface.

Interoperability features:

OSLC automation provider:

- automation request creation (OSLC automation request factory)
  for execution of
  - test case generation
  - test case transformation
  - mutation coverage analysis
  - test script export to version management system
  - test script import for tests developed by test engineer
- automation result retrieval (OSLC query, automation result read request)

OSLC quality management provider:

- test case creation (OSLC test case factory)
- test script creation (OSLC test script factory)
- test execution run creation (OSLC test execution run factory)
- test report creation (OSLC test report factory)
- QM resource queries and read access

OSLC quality management consumer:

- QM resource CRUD access

Delegated user interfaces included/used:

- linked requirements preview (from generated test case)
- linked architecture element preview (from generated test case)

Eclipse integration:

There is no Eclipse integration planned. Management of test case generation tasks is done either via WEFACT or via MoMuT::UML's own web interface.

Availability:

Basic integration (automation provider) is available from project MBAT. More sophisticated aspects need to be developed.

4.5.2 MoMuT::Event-B

MoMuT::Event-B is a new front end for MoMuT to be developed within CRYSTAL. It will share the interoperability features with MoMuT::UML.

4.5.3 Sonar

Sonar is used to evaluate code coverage at TRAIL. An automation provider for Sonar will be developed in order to execute coverage analysis and track coverage analysis results from WEFACT:

Interoperability features:

OSLC automation provider:

- automation request creation (OSLC automation request factory) for execution of
  - test case coverage analysis

Eclipse integration:

n/a

Availability:

Needs to be developed.

4.6 Test Execution

4.6.1 Hudson build and test server

The Hudson build and test server is used to execute JUnit regression tests. Therefore it will also be used to execute the tests generated from the specification model(s). Remote execution of test runs will be provided to be used by WEFACT.

Interoperability features:

OSLC automation provider:

- automation request creation (OSLC automation request factory) for test execution
- automation result retrieval (OSLC query, automation result read request)

Delegated user interfaces included/used:

- n/a

Eclipse integration:

n/a

Availability:

An OSLC automation provider for Hudson is available as open source software.

4.7 Workflow and Traceability

4.7.1 WEFACT

WEFACT manages validation and verification activities and related artefacts. It triggers automatable tasks and tracks traceability links and resource updates.

Interoperability features:

OSLC change management provider

- change request query
- change request read and update

OSLC automation consumer

- automation request creation
- automation plan query
- read automation report

OSLC asset management consumer

- asset query/read

Safety resources consumer

- safety resource read
- safety resource link CRUD

Delegated user interfaces included/used:

- linked resource preview
- automation plan selection (delegated UI or OSLC query)
- asset selection (delegated UI or OSLC query)

Eclipse integration:

Via Mylyn OSLC change management consumer.

Availability:

Automation and asset management consumers are available from MBAT. Other specification parts need to be developed.

An additional specification, currently under development but helpful for optimizing the interoperability, will be Tracked Resource Sets[7]. When adopting it, WEFACT will be able to recognize changed resources more easily.

4.7.2 Mylyn task management front end

Issues are managed at TRAIL with the JIRA issue tracking system. To allow developers a single view on their tasks, integration of JIRA and WEFACT into Eclipse is envisioned, using the Mylyn task management plugin for Eclipse, which can connect to multiple task repositories, including Jira and OSLC change management.

Interoperability features:

[7] http://open-services.net/wiki/core/TrackedResourceSet-2.0/

OSLC change management client

- change request query
- change request read and update

Eclipse integration:

Mylyn is an Eclipse plugin.

Availability:

Mylyn OSLC CM integration is available as open source software.

5 Terms, Abbreviations and Definitions

Please add additional terms, abbreviations and definitions for your deliverable.

AM	Architecture Management
CRYSTAL	**CR**itical S**YST**em Engineering **A**cce**L**eration
CO	Confidential, only for members of the consortium (including the JU).
CRUD	Create/Reade/Update/Delete
D	Demonstrator
O	Other
P	Prototype
PP	Restricted to other program participants (including the JU).
PU	Public
QM	Quality Management
R	Report
RE	Restricted to a group specified by the consortium (including the JU).
RM	Requirements Management
SP	Subproject
TAS PLF	Abbreviation for “TAS Control Platform”
TCG	Test Case Generator
UC	Use Case
V&V	Verification and Validation
WP	Work Package

Table 5-1: Terms, Abbreviations and Definitions