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 SYSTem Engineering AcceLeration

Prototyping IOS concepts D401.021



DOCUMENT INFORMATION

Project	CRYSTAL
Grant Agreement No.	ARTEMIS-2012-1-332830
Deliverable Title	Prototyping IOS concepts
Deliverable No.	D401.021
Dissemination Level	СО
Nature	D
Document Version	V1.00
Date	2014-02-28
Contact	R. Albers
Organization	Philips
Phone	+31 402763212
E-Mail	r.albers@philips.com

Table 1 Document information



AUTHORS TABLE

Name	Company	E-Mail
R. Albers	Philips healthcare	r.albers@philips.com
E. Korff de Gidts	Philips healthcare	eric.korff.de.gidts@philips.com

Table 2 Authors table

CHANGE HISTORY

Version	Date	Reason for Change	Pages Affected
1.0	24/2/2014	Initial version	

Table 3 Change history



CONTENT

1	INTROD	DUCTION	6
1. 1. 1.	1 ROL 2 REL 3 STR	E OF DELIVERABLE ATIONSHIP TO OTHER CRYSTAL DOCUMENTS RUCTURE OF THIS DOCUMENT	
2	OVERV	IEW OF THE IOS PROTOTYPE	7
2. 2.	1 Ver 2 Ver	RIFYING REQUIREMENTS WITH CALIBER AND QUALITYCENTER	7 8
3	DESCR	IPTION OF THE TOOL CHAIN	
3. 3. 3. 3. 3.	1 Bof 2 Pro 3 HP 4 IBM 5 IBM 6 IBM	RLAND CALIBERRM DPRIETARY INTERFACE QUALITY CENTER I DOORS NEXT GEN I QUALITY MANAGER I RATIONAL TEAM CONCERT	11 13 14 17 19 21
4	DESCR	IPTION OF THE USAGE SCENARIOS	
4.	1 ENG 4.1.1 4.1.2 4.1.3 4.1.4	GINEERING METHOD UC_VERIFIYREQUIREMENT Purpose Engineering steps Pre and post conditions Artefacts	25 25 25 25 26 26 26
5	CONCL	USIONS AND WAY AHEAD	
5. 5. 5.	1 LES 2 FEE 5.2.1 5.2.2 5.2.3 5.2.4 5.2.5 5.2.6 5.2.7 5.2.8 5.2.9 5.2.10 3 FUT 5.3.1 5.3.2 5.3.3	SONS LEARNED FROM THIS SPRINT	30 30 30 30 30 30 31 31 31 31 31 31 31 32 32 32 32 32 33 33 33 33
6	TERMS	, ABBREVIATIONS AND DEFINITIONS	



Content of Tables

Figure 1: V-model current process	7
Figure 2 Horizontal traceability in the V-model while utilizing OSLC	9
Figure 3 Separation of concerns in roles and actors	10
Figure 4 Sample screenshot of CaliberRM	12
Figure 5 Sample detailed engineering requirement	12
Figure 6 Baseline content	13
Figure 7 Screenshot UI of proprietary interface	13
Figure 8 Screenshot QualityCenter : identification of releases	14
Figure 9 Screenshot QualityCenter : recognized levels in requirements	14
Figure 10 Screenshot QualityCenter : link requirements to an established test	15
Figure 11 Screenshot QualityCenter : supporting attributes for a given test	15
Figure 12 Screenshot QualityCenter : overview of test results	16
Figure 13 Overview of the Test Traceability Matrix using Doors and QualityCenter	16
Figure 14 Doors Next Gen : representing requirements	17
Figure 15 Details revealed while hovering a requirement (OSLC feature)	18
Figure 16 Visualizing artefact interrelationships	18
Figure 17 UI without hovering	19
Figure 18 UI with information highlighted while hovering content, includes links	20
Figure 19 Sample on 'Verify Movement'	20
Figure 20 Overall status of a piece of test work	21
Figure 21 Screen capture Team Concert : work breakdown view	21
Figure 22 Screen capture Team Concert : traceability view	22
Figure 23 Team Concert : consistent OSLC based hovering information	23
Figure 24 Team Concert : search engine	24
Figure 25 Artefact UML diagram	27

Content of Figures

Table 1 Document information	2
Table 2 Authors table	3
Table 3 Change history	3
Table 4 Affinity between unmet interoperability needs and engineering feedback	
Table 5 Terms, Abbreviations and Definitions	34



1 Introduction

1.1 Role of deliverable

To support development of engineering methods in the Healthcare domain an extra deliverable has been planned on Month 9 of the Crystal project. This deliverable is a simplified example of how an engineering method should be setup, including the use of templates and describing artefacts resulting in a demonstrable product. The delivered parts (documents, templates etc.) will go through the official review process in order to identify any process areas that need extra clarification. Other engineering methods are then able to follow this process in a "first time right" approach.

As this is an example, a light weight engineering method is selected that is common in engineering environments; Verify requirements. It is common as verifying a requirement makes use of the whole V-model and can be applied on many different levels of detail and scopes, extended with specific organizational processes and thus the needs for interoperability's.

1.2 Relationship to other CRYSTAL Documents

This deliverable is a simplified example of how an engineering method should be setup. Please consult the project archive for more detailed information on individual engineering methods.

1.3 Structure of this document

The first part of this document will focus on background information needed to understand choices and company specific processes. In the second part this document will dive into the engineering method and related deliverables, eventually concluding with unmet needs. The unmet needs will be related to the verify requirement engineering method and translated into a first set of conceptual IOS needs for development in the Crystal project.



2 Overview of the IOS Prototype

2.1 Verifying requirements with Caliber and QualityCenter



Philips Healthcare links clinical expertise with human insights to create solutions that bring added value to the entire healthcare cycle - from preventing disease to screening, diagnostics, treatment and aftercare - at home as well as in the hospital. The BIU iXR develops and maintains minimally invasive X-ray solutions that offer diagnosis and treatment of cardiovascular disease.

Our safety critical systems are developed in a regulated environment with high quality demands, extensive legislation and audits. To support these high quality standards Philips works according the V-model in which several different layers are defined related to different parts of the system:



Figure 1: V-model current process

In the first step, left side of the V-model, User needs will be gathered, focussing on the customer's whishes. In the opposite of User Needs in the V-model Validation is executed to make sure Philips creates the right product. In the second step System Requirements are specified, detailing system requirements that have a higher level of detail as the User needs have and also have a development focus. System requirements are tested in the Verification step in the opposite arm of the V-model. Verification shows the right product has been correctly developed and is working according definitions. The next steps of the v-model continue on the above approach until smallest unit parts have been reached.

As soon as the product has been created (whole V model has been worked through) regulating bodies are informed of the new product (submission) and clearance is requested to market the product.

Version	Nature	Date	Page
V1.0	R	2014-02-28	7 of 34



Each phase in the V-model is supported with different tooling and applications. Because of the relation between the left arm of the V-model and the right arm of the v-model a need for interoperability for tools with different purposes is needed. The high level Philips process is as follows:

Requirements are created and maintained in Borland Caliber and serve as the starting point for all projects. Whenever a solid starting point in Caliber is created a baseline is created. This baseline will serve as input for the inhouse developed interface for uploading the requirements and the preselected content into HP QualityCenter. The interface performs various content checks on the baseline of Caliber, before reshuffeling the data into the export format that is compliant for QualityCenter.

The interface checks if values defined in Caliber also exist in QualityCenter such as product names, level, priority etc. When the values are missing in QualityCenter the interface will not start the export of the requirements to QualityCenter but provides warnings. The warnings specify the values used in Caliber but which are not available in QualityCenter, this needs to be solved first.

In QualityCenter the values are added for each warning provided by the interface (error prone, redundant work and project related). Finally able to export the content from Caliber to QualityCenter.

QualityCenter is used to create test cases, manage test execution and result gathering. This leads to specific reports to support the information sharing to regulating bodies such as a Test Traceability Matrix which indicates the link between a requirement and the test case that covers the requirement and its content.

2.2 Verifying requirements with IBM Doors Next Gen and Quality Manager

In the first engineering method, Verifying Requirements, the IBM toolset has been used to prototype interoperability as a "desired state". The same underlying case has been implemented in the IBM toolset as in the setup described above with Caliber and Quality Center. The IBM toolset is based on the Jazz technology, which is a middleware layer common to a number of IBM systems and software engineering tools. The IBM Jazz[™] toolset among other provides a set of OSLC interfaces (Open Service for Lifecycle Collaboration, an OASIS standard). By virtue of this OSLC/Jazz based integration the IBM tools allow for a more flexible and dynamic configuration of the V-model, since no data is copied or synchronised. OSLC implements the Linked Data concept (see e.g. http://en.wikipedia.org/wiki/Linked_data). Of course, the same rigorous validation and verification constraints hold true in any configuration of the V/model of the systems engineering lifecycle. The "desired state" demo with IBM Jazz based tools intends to show how this is indeed possible.

Requirements in this demonstration are defined, both at the level of User Needs and System Requirements, in IBM Doors Next Generation (DNG). The DNG tool is designed to capture, trace, analyze and manage requirements while maintaining compliance with industry standards and regulations. Built using IBM Jazz[™] technology on the team server, DOORS Next Generation provides a single platform for global team collaboration and support for managing requirements effectively, sharing common administration of users, servers and projects

Version	Nature	Date	Page
V1.0	R	2014-02-28	8 of 34



DNG comes with pre-defined templates for requirement types, attributes, links, validations and other configuration details of a project environment. For this demonstration a simple DNG template has been used, to which the notion of User Needs has been added (as a requirement type with pre-existing attributes), and the notion of System Requirement has been redefined from an existing type, to which custom attributes have been added. From DNG requirements can be linked to amongst others planning and test artefacts.



Figure 2 Horizontal traceability in the V-model while utilizing OSLC

Validation and verification in this demonstration is managed in IBM Rational Quality Manager (RQM). The RQM tool is a collaborative hub for business-driven software and systems quality across virtually any platform and type of testing. RQM helps teams plan and organise their quality work, design and construct test cases and test suites, intergate test engines and external tools via adapters, execute and monitor local and remote tests, as well as link these artefacts to other OSLC based resources outside RQM, like requirements or defects.

As a third element in the tool chain of this demonstration IBM Rational Team Concert (RTC) has been introduced to plan and organise work between multi-disciplinary teams. In follow on work RTC will be used more extensively, but in this case defects and tasks are tracked in RTC and a three-level planning (iteration, release, product) is simulated in RTC. This tool provide features that integrate development project tasks including iteration planning, process definition, change management, defect tracking, source control, build automation, and reporting.

Together, the above mentioned IBM tools provide a basis for building an OSLC based tool chain for healthcare specific safecty critical systems engineering, as well as a demonstration of the verify Requirements engineering method. For the latter aspect we show in the demo how requirements

Version	Nature	Date	Page
V1.0	R	2014-02-28	9 of 34



collected in Doors Next Gen are linked to Test Cases in RQM and Tasks and Defects in RTC. The requirements collection is in the demonstration also linked to an RTC iteration plan and an RQM test plan. By virtue of these links traceability views (planning dependencies, impact and coverage analysis) are easily created in the toolset, again, without copying or synchronising data.



Figure 3 Separation of concerns in roles and actors



3 Description of the tool chain

The tool chain from the User needs until the verification and validation is extensive and not in full scope of this example engineering method, to give an idea:

- Modelling tool
- Simulation tool
- Requirements management tool
- Documentation generation tool
- Design tool
- Interface tool
- Software Development tool
- Test management tool
- Document generation tool
- Archiving tool
- Reviewing tool
- Logistic tool
- Metrics and Dashboard tool
- And so on

To verify a requirement the start point is a requirement with the correct content. To define a requirement with the correct content the need for models and visual representations are needed. To transport the requirement an interface has been created that is copied to the test management tool so that test cases can be defined. Before verification (test execution) can be done an actual system with a certain maturity level and quality is needed hence software, hardware and electronics need to be designed, purchased / developed and implemented into a system. This makes use of tool for reviewing, ordering (logistics), metrics, documentation generation etc.

As this engineering method serves as a simplified example for the Healthcare domain a subset of this tool chain is selected to provide input for interoperability: Requirement Management tool in Borland Caliber, Test Management in HP QualityCenter and a proprietary interface to exchange information.

As an alternative together with IBM an alternative demo is shown containing OSLC features that shows the benefits of an OSLC integrated environment.

3.1 Borland CaliberRM

In Caliber requirements are defined. These requirements use different fields (user defined fields):

- Requirement Name
- Version
- Status
- Priority



- Description
- Type
- Planned product release
- Safety related
- Security impact
- Etc.

💣 <u>D</u> etails 🛛 🎇 iXR 🛛 🕵 Re <u>s</u> pons	sibilities 🛛 🗰 <u>R</u> eferences 🛛 🏪 <u>T</u>	raceability	😰 D <u>i</u> scussion 🛛 🎛 <u>H</u> istory 🗍
Name:			
uns.sys.movement		0	
SRQ-CR130090	Version:	<u>O</u> wner:	–
, <u>S</u> tatus: Draft	Priority: Essential		

Figure 4 Sample screenshot of CaliberRM

Eventually leading to a defined requirement with content:

Information:

The system should move fluently through its workflows considering room layout and size.	*
	-

Figure 5 Sample detailed engineering requirement

All the requirements combined together are saved in a baseline. This baseline secures requirements in their context including the user defined fields:



@ CaliberRM
File Edit View Insert Format Requirement Tools Reports Help
◆ → ∽ ½ № % (M) ^{(M})
Baseline Maintenance
Project: iXR File Edit
iXR-AL System Requirements
i
User Needs Specification Allura, shared - 102888 V7.0
🗊 🖓 System Requirements Specification Allura - 35342 V21.0
Encreta User Interaction Design Flexible Viewing - 119371 V5.0
EUNCTIONAL DETAILS - 119377 VI.3
Viewport Javout Management - 119378 V1 3
IIID Allura FlevScreen ViewPort Enlarge - 110303 V/ 3
UID.Allura.riexScreen.viewPort.xperHD - 119395 V2.1
UID.Allura.FlexScreen.Viewport.Snapshot.All - 119397 V5.4

Figure 6 Baseline content

3.2 **Proprietary Interface**

A custom interace has been developed by Philips that imports the requirements from Caliber into Excel, maps the used fields and its content to QualityCenter and exports it to QualityCenter.

X	□ • (° -	-	-	Autor State	-					CaliberToQC.xlsm
F	ile Home	Insert	Page Layout	Formulas Data	Review	View	Developer	r Add-I	ns	
Fro	M From From ess Web Text Get E	From C Sourc	Dther Existin connect ata	g ons Refresh All + @ Edit Lini Connections	tions A ies ks X↓	AZA Sort	Filter	ear eapply dvanced	Text to R	emove Data Cuplicates Validation ~ Data Tools
	115	- (f _x							
	А			В			С		D	
1	Caliber	server:	CaliberTest					Q	C server:	ALM11Test
2	Calibe	r user:	User						QC user:	User
3	Caliber pas	sword:	***					QC p	assword:	***
4	Caliber projec	t:	Crystal					QC dom	ain:	Crystal
5	Caliber baselin	e:	Crystal_base	ine				QC pro	ject:	Crystal_Test
6								QC alerts	(yes/no):	
7								QC tar	get cycles	
8								QC top fo	older	
9								QC req p	refix	
10				Caliber To QC						
11			nr imports: 2	7						

Figure 7 Screenshot UI of proprietary interface

Version	Nature	Date	Page
V1.0	R	2014-02-28	13 of 34



The same user defined field content from Caliber needs to be set in QualityCenter before the export is executed. This is an issue as most of the custom fields in QualityCenter are a multi select list (not just a string value so automated copy pastes could be developed). The multi select lists are editted in a specific customization window in QualityCenter as shown in the <u>Demo movie Caliber HP QC</u> and <u>Demo movie IBM Doors/RQM</u>. The interface uses the API of Caliber using a specific account and password. The user of the interface needs to identify the caliber project and the correct baseline as well as the correct server, users, domain and project for QualityCenter. During the project multiple baselines will be created containing updates and futher specified features leading to imports to QualityCenter. In QualityCenter work can already be executed on previous baseline content, so the interface has been designed to create alerts for requirements that have been changed since the last baseline import.

3.3 HP Quality Center

When the export from the interface to QualityCenter is completed the verification setup can be started. This starts with defining releases in relation to project milestones (integration, verification & validation) and related to the V-model:



Figure 8 Screenshot QualityCenter : identification of releases

When the releases are defined the different levels of requirements are assigned to the releases:

Requireme	ents Edi	it View	Versions	s Fav	orites	Analy	/sis			
📫 🛍	× 🛙 🕸	1 🖞 🕻	3 T •		E,	(e	₩ -		a,	↑ - \
No Filter D	efined)									
08 🗕 🏲		Name				ect Cove	r Status	Req ID		
	Ξ 昌	Requireme	ents					0		
[+	🚞 Verific	ation					32		
	+	🚞 Valida	tion					58		
	+	🚞 Devel	opment					59		

Figure 9 Screenshot QualityCenter : recognized levels in requirements



In the next step test designs and test cases are created and linked to specific requirements:

Tests Edit View Versions Favorites Analysis			
🖆 🐔 🗙 😫 🎍 🦪 🏹 - 💷 🗐 🖉 🍋 - 🔍			
No Filter Defined	Details Design Steps * Pa	arameters Test Configurations	Attachments Req Cov
UUP Name	💼 Select Req 🔓 💭	R .	·
🕀 📕 🚛 Unattached	Entity Name	Req: Descript	tion
Herein Internet and American Ameri American American A	srs.svs.performance.shutd	The systems start-up time shall b	be no longer then 540
🖃 🔚 Verification		, , , , , , , , , , , , , , , , , , , ,	
 E System.test.imaging 			
 System.test.movement 			
System.test.performance			
System.test.performance.coldrestart			
System.test.performance.shutdowntime			
System.test.performance.startuptime			
System.test.performance.warmrestart			
System.test.reliability			

Figure 10 Screenshot QualityCenter : link requirements to an established test

The test cases use fields from Caliber (such as supported products from Caliber that are mapped to Compatible Products in QualityCenter) but also make use of specific QualityCenter user defined fields for managing the test activities such as status, priority, exection time etc.:

Test Details	• X
Test ID: 23 *Test Name: System.test.performance.shutdowntime *Type: 🗎 MANUAL	
Details Details	
Uberlands ** Descing Steps ** Parameters ** Test Configurations # Atachments ** Req Coverage ** Linked Defeds # Business Models (*) History	
OK Cancel Help	

Figure 11 Screenshot QualityCenter : supporting attributes for a given test

Version	Nature	Date	Page
V1.0	R	2014-02-28	15 of 34



Test execution is started collecting more data for analysis of test and results:

E Test Instance Details						
	🖂 • 🎭 👘 🕨					
Name: [1]system.test.movem	ent.movetopark			Cycle: Verification 1	Test Type: 🗎 MAI	NUAL
Details	X 🗉 🖉 T • 🔃 🖂 • 🍕 I	Continue Manual Run				
P> Execution Settings	Sort By: Exec Date[Descending];Exec Time[I	Descending]				Legend
Attachments Linked Defects	€ E Run ID Run Name	Status Duration	Exec Date Exec Date	kec Time Host	Tester	Draft Run
W History	18 🖻 Run_1-30_11	1 😧 Failed 14	1/30/2014 11:42	49 AM NLYBSTQVP1N	300225003	N
						•
	Comments Report Remarks				1	*
	Step Name Status	Exec Date Exec Time	Conditio Steps Details			
	Precondition 🔗 Passed	1/30/2014 11:42:40 AM	Description: System is powered	on and available for use and start	t position is defined in th	ne system settings.
	Move system to starl X Failed	1/30/2014 11:42:48 AM				
			Expected:			
			Actual:			
	•		F			
		OK Cancel	Help			

Figure 12 Screenshot QualityCenter : overview of test results

The most important field for test execution is the status as that identifies the result of the test execution. When all tests for the verification level have been executed a Test Traceability Matrix is created. The Test Traceability matrix is a proprietary report that combines requirement names, linked Test Designs and with the test cases that cover the requirement and the test execution status.

Requirements		Те	st D	esig	ns	Test Cases	
Name	Result		TD_SRS	TD_UID	TD_UNS	Name	Result
		16	7	ы	4		
UNS.Allura.Live image view	Passed	1			х	Live image view	Passed
UNS.Allura.Viewing navigation	Failed	1			х	Viewing navigation	Failed
UNS.Allura.Image overlay	Passed	1			х	Image overlay	Passed
UNS.Allura.Image Enhancement	Passed	1			x	Image Enhancement	Passed
SRS.Allura.Func.SelectViewingExamination	Passed	1	х			SelectViewingExamination	Passed
SRS.Allura.Func.SelectViewingStudy	Failed	1	x			SelectViewingStudy	Failed
SRS.Allura.Func.NavigateToImageData	Passed	1	х			NavigateToImageData	Passed
SRS.Allura.Func.SwitchDisplayFocus	Passed	1	х			SwitchDisplayFocus	Passed
SRS.Allura.Func.MosaicOverview	Failed	1	x			MosaicOverview	Failed
SRS.Allura.Func.SelectPhysioDisplay	Failed	1	х			SelectPhysioDisplay	Failed
SRS.Allura.Func.Select Heart Beat	Not Covered	1	x				
UID.Allura.FlexScreen.ViewPort.Enlarge	Passed	1		х		ViewPort.Enlarge	Passed
UID.Allura.FlexScreen.ViewPort.Enlarge.Restore	Failed	1		х		ViewPort.Enlarge.Restore	Blocked
UID.Allura.FlexScreen.ViewPort.XperHD	Passed	1		х		ViewPort.XperHD	Passed
UID.Allura.FlexScreen.Viewport.Snapshot	Passed	1		х		Viewport. Snapshot	Passed
UID.Allura.FlexScreen.Viewport.Snapshot.All	Passed	1		х		Viewport.Snapshot.All	Passed

Figure 13 Overview of the Test Traceability Matrix using Doors and QualityCenter



The Test Traceability Matrix is used to verify all requirements are covered correctly and thus verification has been completed according process. This information is in general shared with regulating bodies as evidence of successfully following the defined verification process.

3.4 IBM Doors Next Gen

User Needs and System Requirements are collected in Doors Next Gen as elements of two modules which are hierarchical structures of requirements of a similar type in a document like presentation. Attributes and links can be edited directly with primary requirements information. See illustration below.

d Artifact		E All •		1 III III III III III III III III III I	1	2	Overview		
	a		ID	Contents			377: iXR Product	- USER NEEDS SPECIFICATIO IXR Product - USER NEEDS	N
ews 🤍 🏣 🕶		0 /	384	UNS.SYS.MOVEMENT Information:The system should move fluently through its workflows considering room layout and size.			Project: Team Ownership: Content Folder:	SPECIFICATION Crystal IXR (Requirements) Crystal IXR (Requirements) IXR Product - USER NEEDS Eeb 2: 2014 12:22:10 PM	
			385	UNS.SYS.reliability The system should perform without braking for a lengthy period of time with proper preventive maintenance.			Created By: Modified On: Modified By:	Ton van Velzen Feb 2, 2014, 1:43:00 PM Ton van Velzen	
			386	UNS.SYS.performance The system should perform consistently when closing down, rebooting in a cold and warm state or shutting down.			Is Suspect:	Select a profile	100
			387	UNS.SYS.Imaging Images captured with the system should be clear and need to have sufficient contast for different light situations			Format: Launch URL: Package Info:	Module	
							Priority: Requirement Typ: Severity: Source Tag: Status: Owner: Compliance statu Safety: Security:	Medium E: UNS Unclassified SRQ-CR130090 Draft Th. de Laet S: Not applicable False Ence	

Figure 14 Doors Next Gen : representing requirements

OSLC links between requirements, for instance a User Need being covered by System Requirements are created. See illustration below.



004. uns.sys.	Create Link			- X
	Linking from artifact:	🔊 384: uns.sys.movement		SER NEEDS SPECIFIC
svs.movement	Link type: *	Select the type of link		
,	What to link to: (a) Lin	k to an artifact in: Crystal iXR (Requirements)	 Link to web 	
	Search for:	ts Rows in module:	×	
ation:The system shou	Search for artifacts by ID	or for words in the artifact name:		
	*		ß_	
	Select artifact:	Displaying 2	26-34 of 34 matches	
	397: srs.sys.perform	mance	Filter Display by Folder	
	399: srs.sys.perform	mance.colorestart mance.shutdowntime	📄 🚘 Crystal iXR (Requirements)	
	398: srs.sys.perform	mance.startuptime	🔳 🗁 Engineering Tooling Reqts	
	401: srs.sys.perform	mance.warmrestart	🔳 🧁 iXR-CR Product Reqts. (SRQ-CF	a promo U_
	387: uns.sys.imagi 384: uns.sys.move	ng ment		ser Need Specification
	386: uns.sys.perfor	rmance		ext
	385: uns.sys.reliab	ility		
				edium
				ormal
				RC-CR130090
				Drogroce
			Filter Display By Attribute	
			Filter Display by Tag	<i>8</i> -
			Select from View	
		Previous 1 2 Nex		vement
	Create new artifact		OK Cancel	vement.angulate
			srs.sys.mo	ovement.movetopark
			🗿 393:	ovement movetostart
			515.5y5.11	rementanteropolare

Figure 15 Details revealed while hovering a requirement (OSLC feature)

Also OSLC links with artefacts outside Doors Next Gen are created, for instance to related test cases and implementation tasks. The illustration below gives a visualization of a part of the web of artefacts that is thus created.







3.5 IBM Quality Manager

Once requirements are collected in Doors Next Gen a set of test cases in RQM can be generated, or associated, with the requirements collection, and links between test cases and requirements are established.

45: Verify mov	ement to sta	art position	(<u>)</u>			Ð	8	<u>s</u> •	• •	1	• 🖑	Cancel
Sections 🚽	State:	Draft	Action:	Change State	*							
Summary	Originator: Ton	van Velzen	Owner:	Grav Bachelor	*							
Test Case Design	Deleviter II III		-	any success	_							
Formal Review	Priority: U	assigned	<u> </u>									
Development Items	Description: <	Click here to er	nter a descrip	otion >								
Requirement Links												
Risk Assessment	Summary											
Pre-Condition												
Post-Condition											Qui	ality Task: Cre
Expected Results	Use the them	ne, category an	d function fea	atures to group your	test case	es alor	ng rela	ated ite	ms or	logica	groupin	igs. Weight is a
Test Scripts	measure of e	execution effort	and can be b	ased on tester hour	s or unit	s of wo	ork.					
Test Case Execution Records	s Categories	00										
Attachments	Function:	Unassign	ed	1	-							
Execution Variables	Test Phase:	Integratio	n Test		-							
Show All Sections	Estimate:											
	Wainht	100		Points								



The OSLC links automatically retrieve the linked data and present them in the RQM context using DNG data and presentation logic. The fly out in the illustration below shows some requirement details from DNG within the context of an RQM test case to which it is linked.





Figure 18 UI with information highlighted while hovering content, includes links

Test cases are linked in RQM to e.g. test case execution results, and several other artefacts in RQM.



Figure 19 Sample on 'Verify Movement'

One of the OSLC links from the test case is to the Test Plan which is used to monitor the overall status of a particular piece of test work, and shows in real time the actual status of the test plan against a number of parameters (see the progress bars in the top right corner below).



sa bashboarus ~ Requirem	enis y Planning y	Construction Y	Cab Mana)	gement v Dollas v	Execu	auto eceports:	 Change Requi 	5K3 - Y	
Test Plans >				(20)					
6: iXR Produ	ict (UNS, SRS)	Test Plan	for M9	Demo				· 🔶 🗌 Cano	ei Save
Sections	-					Test Case Ev	ecution (Record)	Dronroes.	
Sections	The State:	raft	Action:	Change State	*	I Case Ex	ecuanii (necora)	rogress.	
Summary	Originator: Ton v	an Velzen	Owner:	Arjen van de W	/ete 🕶	Total: o/o h	Estimated: 0%	Progress: 300	Total:
Business Objectives	Priority: 🖨 Med	ium -	-1			Test Culle For			
Test Objectives			-			Test Suite Exi	ecution (Record)	rogress:	
Formal Review						Total: 0/0 h	Estimated: 0%	Progress:0	Tot
Requirement Collection Li Development Plan Links	Description; IXR	Product (UNS	, SRS) Test	Plan for M9 Demo					
Risk Assessment	Pantonia								
Test Schedules	Summary								B
Test Estimation									
Test Environments	200000000000000000000000000000000000000							Quality Task:	Create
Quality Objectives	Overview of th	e test plan.							
Entry Criteria	Categories								
Exit Criteria	Product:	Unassigner	d	1	-				
Test Suites	Release:	Unassigner	d		-				
Test Cases									
Test Case Execution Reco	ords			******				****	
Resources									
A star when a star when									
Attachments									
Show All Sections									

Figure 20 Overall status of a piece of test work

3.6 IBM Rational Team Concert

The Test Plan is then also linked to the development plan in Rational Team Concert which is used to trach the work of a number of teams against a number of product deliverables. Again the plan is a collection of OSLC artefacvts, workitems, and they can be presented with realtime status in several ways. Below a workbreakdown view is illustrated.

💦 🍕 Crystal iXR (Change Management)	Ton van Velzen 🖓 🍝 🗧
oject Dashboards - Work litems - Plans - Source Control - Builds - Reports -	🗄 👻 Search Work Items
Plans > M9 Iteration Plan	🛱 🚔 🤌 🛄 Auto-Save 💽
20 items: 19 open, 1 closed Ends in: 15 days Plan Details Planned Items Links Snapshots Dashboard Notes	
View As: Work Breakdown Transform (1 them excluded) Type to Filter	🕒 🖭 🗦 🌳 Add Work II
Arjen van de Wetering Closed Items: 1 Open Items: 18	Progress: 0/0] +0 h Estimated:
Cosed Items: 0 Open Items: 1	No Work Estimated Progress: 00 +0 h Estimated
Summary	Effective I Progress Status Rank
Angulation imbalances the table in extreme positions	🗢 New Not Ranked
Ton van Velzen	No Work
exacts interior, o I open interior, o	Progress - Estimat

Figure 21 Screen capture Team Concert : work breakdown view

Version	Nature	Date	Page
V1.0	R	2014-02-28	21 of 34



The same view of the M9 Iteartion Plan in RTC can be switched to a Traceability view, showing the objective of this demonstration, namely links (or the lack of them) between tasks, system requirements, test case and defects. One-to-many relationships are possible, and custom markers highlightcertain potential issues in some rows.

🔯 M9 Iteration Plan		(Z)	🛱 🛱 😽 🔲 Auto-Save 🛛 Save
20 items: 19 open, 1 closed Ends in: 15 days			
 Plan Details 			Edit
Planned Items ? Links Snapshots Dashboard	Notes		
View As: Traceability View As: Traceability	 ✓ (2 items excluded) Type to Filter 		🖹 🗄 🗄 🖗 Add Work Item •
Summary	Implements Requirement	Tested By Test Case	Affected by Defect
Implement srs.sys.movement.movetostart	srs.sys.movement.movetostart	8 45: Verify movement to start position	·
Implement srs.sys.imaging	Srs.sys.imaging	寝 Links (3): 1, 2, 3	Bo
🖾 🕼 🖾 Implement srs.sys.imaging.adjustbright	t 🍓 srs.sys.imaging.adjustbrightness	😸 50: Verify brightness adjustment	🐯 123: Brightness adjustmen
Implement srs.sys.performance	Srs.sys.performance	E Links (4): 1, 2, 3, 4	8 0
C (C Missing) Implement srs.sys.movement	Srs.sys.movement		Bo
🖺 Implement srs.sys.imaging.adjustcontrast	srs.sys.imaging.adjustcontrastthe system shall b	🛃 51: Verify contrast adjustment	8 5 -
🖾 🕼 🖬 🖾 🖾 🖾	srs.sys.movement.angulate	😸 48: Verify angular movement (angulate)	8 124: Angulation imbalance
Implement srs.sys.imaging.createimage	srs.sys.imaging.createimageThe system shall be	8 49: Verify crteate image	8 0 -
🔟 Implement srs.sys.movement.movetopark	srs.sys.movement.movetopark	🛃 45: Verify movement to start position	Bo
TC Missing Receap Implement uns.sys.reliabilit	2 🖏 -		8 -
Implement srs.sys.performance.coldrestart	srs.sys.performance.coldrestartThe systems cold	穝 43: Verify performance of cold restart	
🛐 📧 Implement uns.sys.imaging		8 Links (3): 1, 2, 3	
Implement srs.sys.performance.shutdownt	srs.sys.performance.shutdowntimeThe systems	🛃 42: Verify performance shutdowntime	8 0 -
Implement uns.sys.movement	uns.sys.movement	E Links (4): 1, 2, 3, 4	8 0 -
Implement srs.sys.performance.startuptime	srs.sys.performance.startuptimeThe systems sta	🗞 41: Verify startup time performance	Bo
TC Missing (Ret.Gap) Implement-uns.sys.perform.	: 🛃		
Implement srs.sys.movement.rotate	srs.sys.movement.rotate	47: Verify rotation movement	Bo
Implement srs.sys.performance.warmrestart	srs.sys.performance.warmrestartThe systems wa	44: Verify performance of warm restart	

Figure 22 Screen capture Team Concert : traceability view

OSLC linking shows in a similar uniform way as before details of an OSLC artefact in the current context.



🏷 M9 Iteration Plan			[7]	🚉 📮 🔗 🔲 Auto-Save Save
20 items: 19	open, 1 closed Ends in: 15 days			
Plan Details	8			Edit
Planned Item:	s 👔 Links Snapshots Dashboard	d Notes		
View As	: Traceability 🔹 📷 - 🛄 🛱 Excl	ude - (2 items excluded) Type to Filter		🖹 🗄 📄 🛉 Add Work Item -
	Summary	Implements Requirement	Tested By Test Case	Affected by Defect
	Implement srs.sys.movement.movetostar	t 😰 393: srs.sys.movement.movetostart	2 45: Verify movement to start position	8 0
	Implement srs.sys.imaging	402: srs.sys.imaging	🛃 Links (3): 1, 2, 3	8 0
II 🛛 🗳 🔹	🔊 💽 🖾 Implement srs.sys.imaging.adjust	I 123: Brightness adjustment blurs im	age	123: Brightness adjustmen
	Implement srs.sys.performance		age	5 - Affected by Defec
	Cremsing Implement srs.sys.movement	Status: A New		p =
	📕 Implement srs.sys.imaging.adjustcontra	Resolution: Unresolved		3
	🛓 🕼 💼 🕹 Implement srs.sys.movement.ang	Type: Defect	Tags:	124: Angulation imbalance
	🛓 Implement srs.sys.imaging.createimage	Filed Against: Crystal IXR (Change Management)	Owned By: Arjen van de Wetering	5-
	📓 Implement srs.sys.movement.movetopa	Severity: O Critical	Priority: 🗧 Medium	3
	(TEI MISSING) Ret Cap Implement uns.sys.rel	Found In: Unassigned Project Area: Crystal IXR (Change Management)	Planned For: Sprint 1 (1.0) Estimate:	5
	Implement srs.sys.performance.coldres	Creation Date: February 2, 2014 11:13 AM	Time Spent:	a
	🛐 🛪 🗛 Implement uns.sys.imaging	Created By: Ton van Velzen	Due Date: Unassigned	3
	Implement srs.sys.performance.shutdov	Quick Information		p
	Implement uns.sys.movement	Subscribers (1): TvV Refects Plan Item (1): 1		5
	📧 Implement srs.sys.performance.startupti	Description	the brightness unclear questly hour	3 -
	TC Masses) Ret.Gap Implement uns.sys.pe	it happens.	ore originaless, unclear exactly now	a
	Implement srs.sys.movement.rotate	* Show More		
	Implement srs.sys.performance.warmres	art 😰 401: srs.sys.performance.warmrestart	44: Verify performance of warm restart	80 -

Figure 23 Team Concert : consistent OSLC based hovering information

Manipulating links between elements in the view can be done with in-line editing.

	Implement srs.sys.performance.coldrestart	400: srs.sys.performance.coldrestart	43: Verify performance of cold restart	8 0
1 🗆 🗳 •	net.cap Implement uns.sys.imaging		Enks (3): 1, 2, 3	25 -
	Implement srs.sys.performance.shutdownt	Add Explements Requirement htime	42: Verify performance shutdowntime	R o -

The appropriate selection dialogue from DNG is invoked (in this case) to search for the right requirement(s) to link to this row.



quirement Selection	٤
arch for: Artifacts Rows in module:	*
arch for artifacts by ID or for words in the artifact name:	
arch	
lect artifact: Displaying 1-4 of 4 matches (1 a	rtifact selected)
187: uns.sys.imaging	Filter Display by Folder
84: uns.evs.movement 86: uns.svs.performance	🖃 🚖 Crystal iXR (Requirements)
85: uns.sys.reliability	💿 🗁 Engineering Tooling Reqts
	🖃 🗁 iXR-CR Product Reqts. (SRQ-CR
	Product
	🗁 iXR Product - SYSTEM RE
	➢ iXR Product - USER NEEI
	Filter Display By Attribute
	Filter Display by Tag

Figure 24 Team Concert : search engine

This concludes this desired state demo of an OSLC/Jazz based approach to Verify Requirements.

Jazz-	
TEANISERVER	
The Jazz Team Server at sseserver.a	mstec.ibm.com in Jazz requires a user ID and password.
000	User ID:
	Password:
	Remember my User ID
Licensed Material - Property of IBM Corp. © Copyright IBM logo, Jazz, and Rational are tradiemarks of IBM Corporatio Eclipse is a trademark of Eclipse Foundation, Inc. Java an trademarks of Oracle and/or its affiliates in the United Stat	I Corp. and its licensors 2008, 2013. All Rights Reserved. IBM, the IBM n, in the United States, other countries and regions, or both. Built on d all Java-based trademarks and logos are trademarks or registered es, other countries and regions, or both.
TRM	Rational. software



4 Description of the usage scenarios

4.1 Engineering Method UC_VerifiyRequirement

4.1.1 Purpose

The objective of this engineering method is to provide a clear and condensed overview of applicable requirements, associated tests, the outcome of the tests, and - derived from this - the engineering status of a work product. The matrix can be used in the engineering life cycle once the engineering requirements are established and the associated test objectives are identified. The matrix highlights any unfinished or problematic engineering requirement, as it backtracks the outcome of a test(s) to their originating engineering requirement(s), but it can also be used for engineering progress tracking (assuming that tests are available prior to the actual implementation). The matrix is depending on proper requirements will pass unnoticed. When supported with by automated regression testing, it can be particularly useful for iterative engineering approaches, as it keeps track on the status of any engineering requirement outside the scope of a particular iteration. Once incorporated into a report, the matrix is formally reviewed and archived as supporting evidence at the milestone gate review of the subsequent next phase in the engineering life cycle.

4.1.2 Engineering steps

The engineering method "verify requirement" consists out of the following engineering steps;

1. Create requirements in an application for Requirements Lifecycle Management (RLCM) and make a requirements document available in the Application for managing documentation lifecycles (PLDM).

1a. The OSLC interface propagates the approval status and traceability between requirements coming from PLDM to any other engineering application.

2. In the Test Management & Execution application (TMAE) the requirement can directly be seen with its content and its relation to other lifecycle artefacts (e.g. with requirements).

3. The RLCM and TMAE environment can generate a report that highlights all applicable requirements. Both applications can handle the same template makeup.

4. A Test Design is created for a cluster of related requirements; requirements are decomposed into Test Cases while their relationship is set for requirements-to-test traceability purposes (aka. Test Tracability Matrix).

4a. Test Cases are automatically flagged when the content of a requirement changes. The flag indicates the need for a proper impact assessment on the change impact on the Test Case itself.

5. As soon as all Test Designs and Test cases are created, different reports can be generated, such as a traceability matrix, a test design overview, or a test case overview, and they are directly available in PLDM.

6. As soon as approvals in the PLDM application are given triggers are visible in the linked applications.

6a. In the Test management & execution software the test designs and cases are set on status reviewed / approved.

7. Test Execution can be started. As soon as a test case is executed the status of that test case is updated to all linked applications (live status updates).

Version	Nature	Date	Page
V1.0	R	2014-02-28	25 of 34



7a. In RLCM the requirement with its linked test case and including the status is visible and can be reported on.

4.1.3 Pre and post conditions

The engineering method requires the following pre-conditions to be present in order to proceed;

- Applications that can share data in its context
- Requirements are available
- Tests are available
- Test Results are available (optional)
- Defects are known (optional)
- Relations between requirements and test cases are defined
- Rework is pending (hence a need for changes)

The end stage of this engineering method results the following post conditions:

- The engineering requirement(s) are verified
- There is an authorized verification report containing the verified requirement(s)
- A Traceability matrix is available

There is no manual push and pull interfaces and extra manual checks on data integrity and consistency.

4.1.4 Artefacts

The figure below depicts an UML representation of the engineering artefacts required for this engineering method. The sub-paragraphs elaborate further on the artefacts mentioned.





UML class representation artifacts E. Korff de Gidts rev 0.3

Figure 25 Artefact UML diagram



4.1.4.1 Artefact - Requirement

Individual requirement posed to the product under development, stating a desired characteristic of the product or services. Includes functional or performance requirements (ISO). The requirements can be organized and viewed into logical and/or hierarchical sub-groups. The description is as rich as hypertext, thus allows for e.g. tables, mathematical formulates, references, multimedia content, illustrations, or even interactive simulation. To ensure on the long term availability, content can be easily copied, extracted, or uploaded from generally available editors (like word or excel), publishing tools, or web servers. Multiple artifacts, their meta data, and trace relations and can be extracted from the import against a set of custom rules. This to enhance the easy of repeatability/reproducibility and to avoid a laborious and error prone two-stage approach.

Shared properties:

- Requirement headline
- Requirement reference ID
- Requirement description
- Requirement category tags
- Requirement author(s)
- Lifecycle status information

4.1.4.2 Artefact - Test

An individual test, setup to verify one or more characteristics of a product or service. It is a sequence of actions and checks, where the actions indicate the test stimuli while the checks highlight the expected response. Its characteristics match that of the 'Requirement' artefact.

For the ease of automated regression testing, an action and/or check may refer to one or more test scripts that automate (part of the) procedure. Custom rule sets utilizing meta data allow for the sequencing of tests into a test run, scheduled for a particular test environment (while assuming that the test environment is suited for that particular test).

Shared properties:

- Test headline
- Test reference ID
- Test description
- Test category tags
- Test author(s)
- Lifecycle status information

4.1.4.3 Artefact – Test Result

Individual outcome of a test, capturing the outcome of the test, any deviations from the (formal) test procedure, and/or particular remarks or observations made. Its characteristics match that of the 'Requirement' artefact.

Shared properties:

- Test date/time
- Test reference ID
- Test environment ID
- Test outcome
- Test deviations
- Test remarks
- Test engineer(s)



4.1.4.4 Artefact - Document

Individual document or report, consisting out of one or more files in formats supported by the engineering environment. Information is subject to access and change control.

Shared properties:

- Document title
- Document type
- Document reference ID
- Document author(s)
- Lifecycle status information

4.1.4.5 Artefact – Traceability matrix

Compiled overview where, for each requirement applicable, the outcome of the associated tests are interpreted and translated into a requirement outcome following a custom rule set. The enumerated states and their associated conditions can be defined freely (and typically takes meta data like the lifecycle status of the underlying information into consideration).

Shared properties:

- Req. headline
- Req. outcome
- Test headline
- Test outcome
- Req. outcome status info



5 Conclusions and way ahead

The Engineering method UC_VerifiyRequirement acts as a pilot for exploring the way of working for charting other engineering methods. More engineering methods will follow, where the lessons learned from this engineering method will be taken into consideration.

5.1 Lessons learned from this sprint

- Class Diagrams are an effective means for describing the interrelationship between the various artefacts.
- Industry partners and engineering institutes lack a common engineering language. Analysis of state-ofthe-art ontology shows that additional effort is needed to implement IOS ontology (1st) and explore the possibility to generalize upon a shared and generic engineering ontology (2nd).
- Incorporating the lessons learned from CESAR and MBAT into the work packages of CRYSTAL is not a trivial exercise.

5.2 Feedback from engineering teams

Charting the current way of working of the UC_VerifiyRequirement engineering method revealed various unmet needs in the current engineering tool environment (CaliberRM + HP Quality Center). The main observations, as received from the engineers involved, are elaborated upon here. This provides the bases for the high-level abstract in terms of tool interoperability in paragraph 5.3.

5.2.1 Baseline size

One of the current problems we discovered in our engineering tooling relates to the baseline functionality of CaliberRM (but it also applies to tooling from other vendors)

From a regulatory perspective, any particular platform release has to provide a complete and consistent set of records on the development, design, verification, and validation effort, all in line with the characteristics of the product released. In addition, we seek to emphasize on the similarities in intended use between the new or modified platform release and the earlier release that was granted market approval by the regulatory authorities.

Various business incentives thus favor the management of engineering requirements at platform level (as also depicted in the UML diagram of the Public Use Case Healthcare Verify Requirement).

Proper baseline functionality is required in order to manage the ~ 15.000 engineering requirements that make up the platform, its commercial variations/configuration, and subcomponents. Especially since various engineering teams are working in parallel in a project setting to prepare for a new platform release or the development of a particular subcomponent. The current engineering tooling environment can manage up to ~ 3500 engineering requirements. Requirements are thus currently centered on a theme or subcomponent in order to deal with this limitation. This situation is however not ideal as it hampers traceability, reporting, or material referencing.

In this requirement structure, a cluster of projects is created to handle a volume of 15.000 engineering requirements.

5.2.2 Proprietary tooling

CaliberRM is currently in use for Requirements Life Cycle Management (RLCM). HP Quality Center is used for Test Management and Execution (TMAE).

These engineering tools mentioned lack a level of interoperability that fits Philips engineering needs. Philips developed a proprietary tool that satisfies these need, but this tool comes with a burden for tool validation (as this is required by regulatory bodies) and a cost of ownership that Philips seeks to avoid. This requirement shows the main characteristics of this proprietary tool.

Version	Nature	Date	Page
V1.0	R	2014-02-28	30 of 34



5.2.3 Object Linking and Embedding (OLE)

Each individual requirement posed to the product under development states a desired characteristic of the product or services, includes functional or performance requirements (ISO). A requirement description can be as rich as hypertext, thus allows for e.g. tables, mathematical formulates, references, multimedia content, illustrations, or even interactive simulation. In order to ensure on the long term availability of this content and material, it is preferred that this kind of content can be easily copied, extracted, or uploaded from generally available editors (like word or excel), publishing tools, or web servers.

Once the content is preserved it is handled as any other requirement description, where it can be referenced/incorporated in reports like PowerPoint or word or altered from within the engineering environment.

(e.g. via OLE, the matching editor can be started on the content and the engineering environment is updated with any 'save' operation).

5.2.4 Bitmaps

Closely related to Object Linking and Embedding (OLE) discussion is an unmet need for being able to incorporate and manipulate bitmaps in a requirements description. Basic operations like zoom, pan, crop, rescaling, edit transparency, etc. for high-resolution images are assumed present for managing the content (on import) and/or for presentation purposes (on export when incorporated in a document).

5.2.5 Traceability audits

Closely related to the 'Baseline size' discussion is the unmet need for being able to conduct traceability audits crossing the scope of a 'project'. Most requirement engineering tools recognize the concept of a 'project' or 'binder' or some other type of container that acts as top-level structure under which all relevant engineering requirements are organized. This same top-level structure is also found to be a logical boundary for many of the tools engineering support features.

(e.g. a traceability link has to come from within a 'project' structure, or a traceability matrix can include information originating from this structure)

Being able to relate project to one another may offer a means to cross these boundaries and thus provides the means for conducting traceability audits across multiple 'projects', all within the scope of platform. This would allow for more, instead of fewer but bigger top-level structures, assuming that there are easy means to define something like a soft-links or symbolic link alike in such a structure.

5.2.6 Vendor support

Closely related to the 'Proprietary tooling' discussion is the call for tailoring options and vendor support. Philips, as well as other engineering companies, seeks engineering tools that seamlessly integrate in their organize structure, environment, and way of working. Tools typically fail to meet all criteria posed, hence some level of adaptation is required; both from an organizational perspective, as from within the tools themselves. For the latter part, a company will seek the support of the tool vendor where it has to rely on some level of support for tweaking the tools to their particular needs. Improvements on the current level of tailing options and vendor support is requested.

5.2.7 Information consistency

Most engineering tools consider the requirements engineering environment as the primary means and portal for modifying the requirements content, meta data, and references. Within our organization, requirements are subject to a reviewing process / formal inspection, where the requirements are 'exported' from the tooling environment into a document. The document itself is subject to a document life cycle and archived in a document management system, and is considered the 'authorized' representation of a requirement. Hence the unmet need for information consistency between the content of the document management system and its meta data with the requirements engineering environment, all in a controlled manner that includes change or baseline management.

Version	Nature	Date	Page
V1.0	R	2014-02-28	31 of 34



5.2.8 Traceability matrix editing

The pattern here is more or less closely related to the 'Information consistency' discussion, but now specifically for the produced Traceability matrix. In the current way of working, when a mistake is found in the produced traceability matrix, one needs to find the matching entries in the requirements engineering environment, edit it, repeat the report generation part, and cross-check that the new output matches the desired rework outcome. This is a rather laborious path. Instead it is preferred to edit the outcome, where the requirements engineering environment follows in pursue.

5.2.9 Ease of use

Requirements engineers prefer engineering tools that are effective and efficient and don't require many user interface interactions for accomplishing a certain task. Being able to manipulate a group of requirements all at once helps to boost productivity, especially considering the high volume of requirements at hand. The depth of menus and dialogs, and the amount of mouse clicks and selections too are factors to consider for the ease of use.

5.2.10 Variation management

A platform typically supports various commercial configurations and variations. As such, one seeks a mechanism that allows for the identification of all relevant engineering requirements that match the individual product makeup. E.g. an X-ray product may support 3 table types, 2 detector types, and some optional components, while other configurations aren't supported within that particular product. Which engineering requirements do apply for this configuration? (Note that some configurations can be mutually exclusive, influence one another, or depend on each other)

5.3 Future work

While interpreting the unmet needs from 5.2, some high-level conceptual tool interoperability issues were recognized. The table below shows the affinity and interrelationship between these two, thus gives an indication of their relative weight and contribution.

The CRYSTAL IOS is expected to be a major enabler to design a solution for the following unmet needs.

			Feedback from engineering teams								
		Baseline size (§5.2.1)	Proprietary tooling (§5.2.2)	Object Linking and Embedding (§5.2.3)	Bitmaps (§5.2.4)	Fraceability audits (§5.2.5)	Vendor support (§5.2.6)	nformation consistency (§5.2.7)	Fraceability matrix editing (§5.2.8)	Ease of use (§5.2.9)	Variation management (§5.2.10)
ed	Variation management (§5.3.1):		A					A			
net ne	Rich content (§5.3.2)										
Νυ	Information consistency (§5.3.3)										

Table 4 Affinity between unmet interoperability needs and engineering feedback



5.3.1 Variation management (User Story 2.03)

The need for variation management covers various related themes of needs;

- A platform typically supports various commercial configurations and variations. As such, one seeks a
 mechanism that allows for managing the variation points and common platform features within the
 product family, along with their dependencies and constraints. Ultimately this will allow for 'configuring' a
 particular new product built while re-using the common features of the product platform. Once the
 configuration is defined, it would allow for the identification of all relevant engineering requirements,
 associated tests, or other artefacts relevant, which would aid the regulatory needs for having a
 consistent set of product documentation and verification/validation evidence for a given product release.
- While engineering requirements are managed at product platform level, concurrent engineering teams are active in (re)defining some artefacts within the scope of their project assignment. Their project specific contribution has to be split between 'generalized' contributions and 'configuration specific' contributions, but can also conflict with engineering decisions made by the other concurrent teams. Within the software engineering realm various configuration management tools offer support for branches and/or (auto) merging source code lines. A similar concept is appreciated.
- Most engineering tools recognize some 'top-level' structure with supporting meta data (e.g. a project with its engineering team members and the associated roles and permissions). This 'top-level' structure typically also acts as natural 'boundary' for e.g. engineering reports, cross-referenced information, or a search function. While more complex products typically have volumes of information, one seeks to partition this information across logical and/or architectural compositions where details are added the deeper one goes into the structure. It would be ideal when the presence of a new top-level structure is propagated across the eco systems of engineering tools used (instead of having to manually add a new project in any of the engineering tools used), while the volume data can be partitions (e.g. soft-link alike).
- Variation management extends to DSLs and/or models used for engineering that particular product release. As variation management is not yet fully supported, one can easily run into inconsistencies or erroneous model assumptions.

5.3.2 Rich content

Most engineering tools use text as their predominant format description format. While a natural language can be descriptive, alternatives like tables, mathematical formulates, references, multimedia content, illustrations, or even interactive simulation can be more powerful means of communication. Support for rich content that is easily imported and/or copy-pasted from generally available editors (like word or excel), publishing tools, or web servers aids the engineering teams and ensure on the long term availability of content and material.

5.3.3 Information consistency (User Story 4.01)

Like for variation management, various needs on information consistency issues were identified;

- Most engineering tools consider a certain engineering environment as a primary means and portal for editing their content or meta data. E.g. CaliberRM for requirements or HP Quality Center for test related artefacts. Artefacts are typically 'exported' from an engineering environment into a document, where the document is considered to be the 'authorized' representation of the incorporated artefacts. The document itself is subject to formal reviews/inspections, has its own document life cycle, and is archived in a document management system as part of regulatory evidence and/or to ensure on the long term availability of product related documentation. As documents are important artefacts one seeks to have their content kept consistent with the underlying engineering environments; e.g. when a document is reworked one would prefer to have the associated engineering environments to follow in pursue, or the opposite around. All in a controlled manner that includes change or baseline management. One thus prefers multi entry points for editing the same content.
- Engineering tools support and preferably automate certain engineering tasks (e.g. visualize or model requirements, generate code via DLSs, or check for consistencies). It is observed that these tools operate in isolation; there is no systematic approach to relate different models or to maintain the consistency between them. Overall information consistency is desired but most likely hard to achieve. A generic way to ensure upon the traceability of information is considered an easier to reach alternative.

Version	Nature	Date	Page
V1.0	R	2014-02-28	33 of 34



6 Terms, Abbreviations and Definitions

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

CRYSTAL	CRitical SYSTem Engineering AcceLeration
R	Report
Р	Prototype
D	Demonstrator
0	Other
PU	Public
PP	Restricted to other program participants (including the JU).
RE	Restricted to a group specified by the consortium (including the JU).
СО	Confidential, only for members of the consortium (including the JU).
WP	Work Package
SP	Subproject

Table 5 Terms, Abbreviations and Definitions