

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.



CRritical **SY**STem Engineering **Acce**Leration

UC 4.1

Medical procedures in an interventional X-ray system

D401.010

DOCUMENT INFORMATION

Project	CRYSTAL
Grant Agreement No.	ARTEMIS-2012-1-332830
Deliverable Title	Medical procedures in an interventional X-ray system
Deliverable No.	D401.010
Dissemination Level	CO
Confidentiality	R
Document Version	1.0
Date	2013-11-11
Contact	Rob Albers
Organization	Philips Healthcare
Phone	+31-402763212
E-Mail	R.Albers@philips.com

AUTHORS TABLE

Name	Company	E-Mail
Rob Albers	Philips Healthcare	r.albers@philips.com
Thomas de Laet	Philips Healthcare	Thomas.de.laet@philips.com

CHANGE HISTORY

Version	Date	Reason for Change	Pages Affected
1	2013-11-11	Initial version	

CONTENT

1	INTRODUCTION.....	5
1.1	ROLE OF DELIVERABLE	5
1.2	RELATIONSHIP TO OTHER CRYSTAL DOCUMENTS	5
1.3	STRUCTURE OF THIS DOCUMENT	5
2	HIGH LEVEL DESCRIPTION OF USE CASE AND CONTEXT	6
2.1	RATIONALES.....	6
3	BUSINESS NEEDS FOR WORK PACKAGE 4.1	9
4	ENGINEERING METHODS	12
4.1	SCENARIO 1: ORIENTATION OF X-RAY IMAGE ON MONITOR.....	13
4.1.1	<i>User Needs</i>	13
4.1.2	<i>Case study description</i>	13
4.2	SCENARIO 2: SETTING X-RAY BEAM PROJECTION WITH A JOYSTICK	14
4.2.1	<i>User needs:</i>	14
4.2.2	<i>case study description:</i>	14
4.3	SCENARIO 3: MOVEMENT DIRECTION OF BOLUS CHASE.....	14
4.3.1	<i>user needs:</i>	14
4.3.2	<i>case study description:</i>	14
5	ENGINEERING METHODS DETAILED DESCRIPTIONS.....	15
5.1.1	<i>Requirements verification</i>	15
6	TERMS, ABBREVIATIONS AND DEFINITIONS	17
7	REFERENCES.....	18
8	ANNEX I: DETAILED DESCRIPTIONS OF THE ENGINEERING METHODS	19
9	ANNEX II: TECHNOLOGY BASE LINE & PROGRESS BEYOND.....	20
10	ANNEX III: VISUAL REPRESENTATION OF CURRENT SITUATION.....	21

1 Introduction

1.1 Role of deliverable

This document has the following major purposes:

- Define of the overall use case, including a detailed description of the underlying development processes and the set of involved process activities and engineering methods
- Provide input to WP601 (IOS Development) required to derive specific IOS-related requirements
- Provide input to WP602 (Platform Builder) required to derive adequate meta models
- Establish the technology baseline with respect to the use-case, and the expected progress beyond (existing functionalities vs. functionalities that are expected to be developed in CRYSTAL)

1.2 Relationship to other CRYSTAL Documents

1.3 Structure of this document

The structure of the document is as follows: Section 2 describes the high level use case and context for work package 4, in which we cover the complete V-model.

WP4.1 focuses on the top-horizontal layers (requirements generation, verification and validation), WP4.3 focuses on the bottom-horizontal layers (model driven development, engineering and testing). Finally WP4.2 focuses on the safety and non-functional requirements across the V-model.

Section 3 describes the business needs for work package 4.1, including a detailed V-model, with on each level the input and output flow of information including the tools that are currently involved,

Subsequently, section 4 contains a list of engineering methods for this work package and three practical user scenarios on which we want to validate the work.

The document concludes with a detailed description of the first engineering method “verify requirement”.

2 High level description of use case and context

2.1 Rationales

Healthcare systems are subject to strict regulations from ISO, IEC and FDA regarding safety of operators and patients [Ref ISO/IEC/FDA norms]. A well-defined development process needs to be defined including harm and hazard analysis, risk management and extensive documentation for that purpose. The development process is typically following the 'traditional' V-model; Figure 1 (left) outlines this V-model while Figure 1(right) maps this onto the documentation.

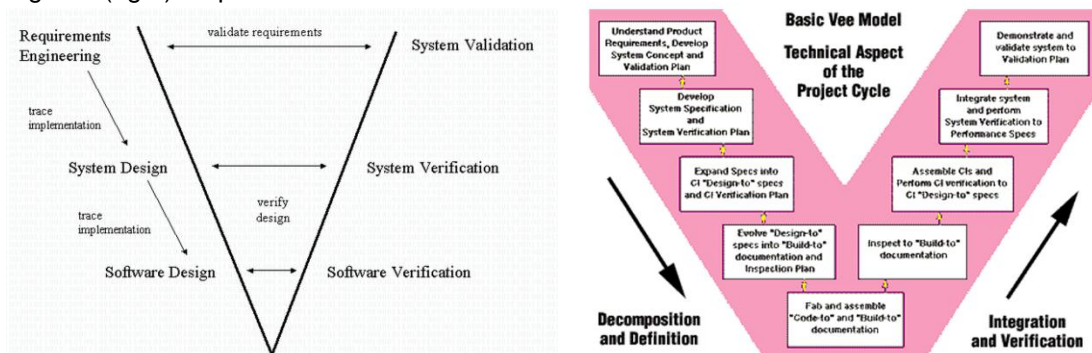


Figure 1: the V-model showing the process (left) and the documentation (right). Pictures are borrowed from internet sources and Mouz et. al. (1996,2000)

V-Model: Advantages of linearly following the V-model, in particular for safety, include the well-documented record and audit-trail of process and products, and the 'push-forward' nature of obtaining the final product, which fits engineers quite well. Among the downsides are a lack of incremental approaches, the late system integration and the extensive documentation (which must be updated upon every change and for every different member of a product family). A particular consequence of the late integration is that negative effects of safety measures on usability are observed only in a very late stage, or even only in the field. In practice this leads to much manual effort in producing documentation and defining tests.

New challenges: Safety-critical systems engineering faces also new challenges. The complexity of systems is ever increasing due to higher customer demands, more advanced functionality and integration with other medical equipment. System components, in particular, software components become COTS rather than proprietary and, since many safety aspects are software defined, new methods are needed for guaranteeing safety for component-based systems. In addition, systems have to be compliant with updated and new regulatory norms. Because of this, and because of error corrections and changing requirements, updates in the field have to be performed. Finally, in order to maintain a competitive edge, time-to-market must be kept as small as possible or at least predictable.

Improvements: Although current systems do satisfy the safety requirements, there is a need to improve on the following aspects:

1. Level of interoperability between applications. For example to support complete requirements traceability to test cases to comply with regulatory (WP 4.1)
2. The development effort and lack of early feedback on extra-functional requirements. (WP 4.1)
3. The call-rate due to a mismatch between user needs and final implementation. (WP 4.2)
4. High release effort due to late integration and manual testing of non-functional (e.g. safety) requirements. (WP 4.3)

The goal of the CRYSTAL project is to improve these four metrics through a change in the engineering process and in the tool support. At the same time these four are the respective drivers of the three use cases of Philips in the healthcare domain in CRYSTAL.

Regarding the process, we require it to be much more iterative and admitting to examine system behaviour and consequences of choices in an early stage. An example of an iterative approach is given in Figure 2, proposed by Barry Boehm as an iterative waterfall in which each iteration provides increasing (software) capabilities [Boehm 1988]. The developed system goes through four cycles:

1. Proof-of-concept cycle — define the business goals, capture the requirements, develop a conceptual design, construct a "proof-of-concept", establish test plans, conduct a risk analysis. Share results with user.
2. First-build cycle — derive system requirements, develop logic design, construct first build, evaluate results. Share results with user.
3. Second-build cycle — derive subsystem requirements, produce physical design, construct second build, evaluate results. Share results with user.
4. Final-build cycle — derive unit requirements, produce final design, construct final build, test all levels. Seek user acceptance.

The entire application is prototyped together with the user and any gaps in requirements are identified into more detail as work progresses. Iterations are then continued until the implementation is finally accepted, conveying very clearly the cyclic nature of the process.

The consequences of an iterative approach on extra-functional properties and in particular on safety are significant. To mention two aspects: there is a lack of a single traceable process (leading to extensive documentation updates during each cycle) and verifying safety properties in this incremental way leads to much more work. **The vision and aim of the CRYSTAL project is to alleviate this problem as well as to improve upon the development metrics through a seamlessly interoperable tooling standard.**

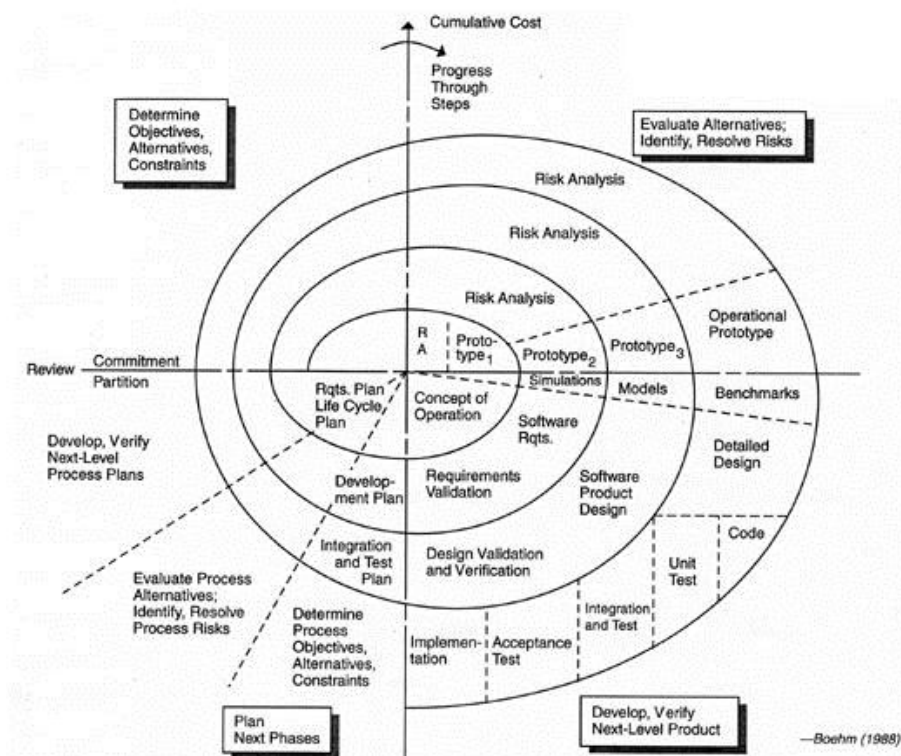


Figure 2 Spiral Development [Boehm 1988]

Regarding tools, these are already used during all phases in system design and implementation, typically with the aim to support and automate certain tasks. Examples are tools for visualizing requirements, for requirements modelling and consistency checking, tools (and languages) for architecture descriptions, and documentation management tools. Important observation is that currently, these tools operate on isolated

aspects of the design and use specific underlying models (if any). *There is no systematic approach yet to relate different models and to maintain consistency between them.*

Characteristics of the approach that CRYSTAL takes are the following:

1. The entire system engineering process is based on a collection of **interoperable models**. These models can be new and specific or models underlying existing (commercial) tools.
2. Models are related by **model transformations**, supported again by **tools**, defining an **InterOperability Specification (IOS)**. Design decisions are also documented as models and transformations.
3. Representations like graphs, figures, schematics, animations and even documentation and simulators are **derived from these models**.
4. Components and system parts are represented in the models through rich interfaces (including extra-functional properties). Simulation tools support the easy switch between actual and simulated system parts.
5. The overall result is a **seamlessly interoperable tool chain** for the support of the system engineering process.

The CRYSTAL Healthcare domain will investigate these tooling and models during the iterative development cycle of safety-critical systems engineering, applied to industrial use cases where patient safety is absolutely critical but the usability of the system should not be compromised. The results are input to a **system engineering tool chain**.

We will use language technology for representation and translation of the models, in particular, *Domain Specific Languages* (DSLs). Domain Specific elements concern the different purposes of the models as well as the application domain. Existing DSL tools will help significantly to define the models, to define transformations and to automate the development of simulators.

3 Business needs for work package 4.1

The previous paragraph described the rationales and improvement metrics for work package 4. We now focus on the two business needs for work package 4.1:

- (1) Level of interoperability between applications to support complete requirements traceability to test cases to comply with regulatory
- (2) The development effort and lack of early feedback on (extra-)functional requirements.

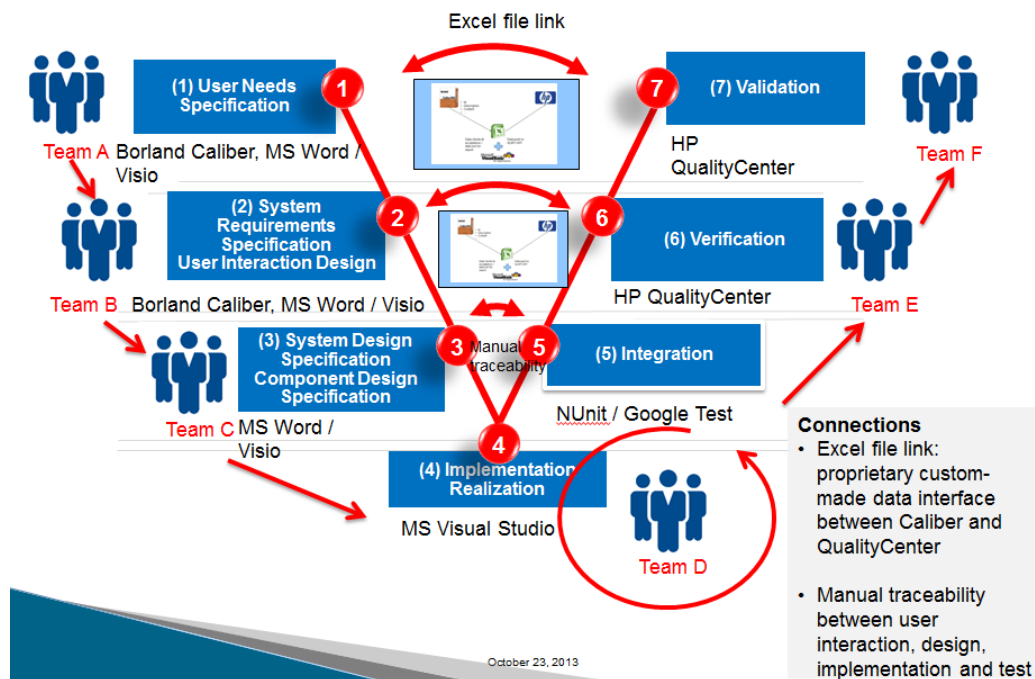


Figure 3 Current development process

In Figure 3, the current development process is shown based on the sequential (V-Model). Following the figure from the top left, starting Team A creating textual User Needs Specifications (UNS), at the end of the phase, a (sequential) handover is planned to Team B of people creating user interactions specifications. Similarly, when the User Interaction Specification (UIS) is finalized, (again) a sequential handover is planned to software development (Team C) where the UIS is input for a system and detailed design specification. Finally, a team of software engineers (Team D) implement the detailed design after the design is finalized in the previous phase. From Implementation phase, the testing phase is started, subsequently followed by verification (Team E) and validation (Team F).

The current process is lacking incremental approaches, gives room for late system integration and extensive documentation (which must be updated upon every change and for every different member of a product family). A particular consequence of the late integration is that negative effects of safety measures and usability are observed only in a very late stage, or even only in the field. In practice this leads to much manual effort in producing documentation and defining tests.

In the table below, the V-model is shown in more detail, with on each level the input and output flow of information. The fourth column is showing the tools that are currently involved.

Version	Confidentiality Level	Date	Page
V1.00	R	2013-11-11	9 of 25

Input		Output	Tools
1. User Need Spec Stakeholders: Doctors, Philips Marketing, Service, Manufacturing		Textual (video?) description, feature list Size: 2 X A4, user understandable Stable, focus on product family Effort: 1 => UNS	Word (file create) Agile DHF (PLM) Caliber (traceability)
1.1 Project definition	User Need Spec Project agreement	Project Agreement, Project Plan Effort: 3	Word (file create) Agile DHF (PLM)
2. System Req Spec Stakeholders: Standards Department	User Need Spec Project agreement	1 doc per Product Release/Instance List of system functions, standards, non-functionals User Understandable, marketing document 100 X A4 Effort: 5 => SRS	Word (file create) Agile DHF (PLM) Caliber (traceability)
2.1 Technical concepts	Project agreement	Impact analysis of PA features 10-50 x A4 Effort: 5 => TC xx	Word (file create) Agile DHF (PLM)
2.2 Master Test Release Plan	Project agreement, Technical concepts	MTRP + project plan for test&integration	Word (file create) Agile DHF (PLM) manual traceability
2.1 User Interaction Design	SRS, System Design, Detailed Design (iterative)	Rational, history of changes User Scenarios => Visual Model of Dynamics → living document, updated regularly UID (User Interaction Design): state behavior, workflow description restrictive specs. system as a black box 1000 pages Effort: 10	Word (file create) Agile DHF (PLM) Caliber (traceability)
3.1 System Design	SRS, UID	System Design: Architectural design (decomposition) Functional Analysis, component interfaces Component Interaction, Component behavior - states - activities SDS (System Design Specification) 200 pages minimum Executable models Effort: 2	Word (file create) Agile DHF (PLM) manual traceability
3.2 Component Design	System Design, UID	Component Design: Same as System Design, but on comp. Level Executable models 50 pages maximum Test plan Effort: 20	Word (file create) Agile DHF (PLM) manual traceability
4 Implementation/Realization	Component Design	Software, Electronics, Mechanics Effort: 50	ClearCase (SW) ClearQuest (defects) manual traceability

5.1 Integration	Implementation, Test plan, Test scripts all left side output Traceability to left hand side	Integration and Test designs - no traceability Effort: 50	Word (file create) Nunit / Gtest (SW test) QualityCenter (PLM) manual/specific Excel interface traceability
6. Verification	SRS, UID -> TestDesign / Testcases	Verification Test report Traceability matrix to SRS Tracking sheet traceability to UID Effort: 40	Caliber (input req) specific Excel interface traceability QualityCenter (PLM) ClearQuest (defects) Agile DHF (PLM)
7. Validation	User Needs Specification -> User Needs TestDesign / Testcases Customer input	Validation report Validation Traceability Matrix to UNS Effort: 20	Caliber (input req) Word (file create) QualityCenter (PLM) manual traceability Agile DHF (PLM)

A first step in defining an incremental development process is moving away from multiple mono-disciplinary component teams towards a single multi-disciplinary system team.

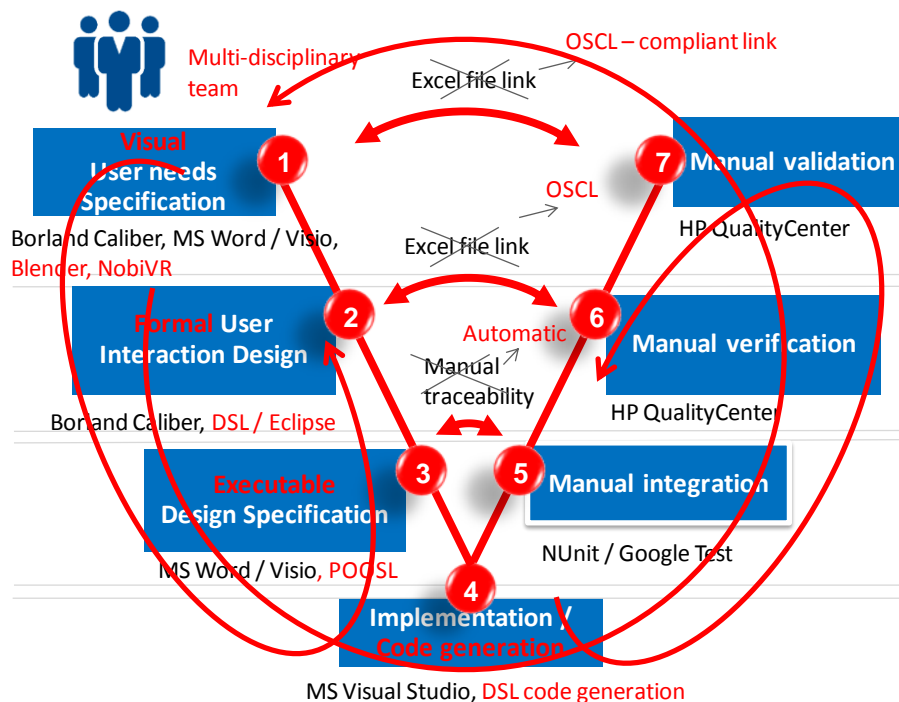


Figure 4 Proposed incremental development process and connections

Regarding tool support, UNS specifications are strictly defined but tools are lacking for complete traceability up to implementation layer which may result in changing behavior defined in later stages, creating a possible gap between user needs, system specification and final implementation. A second step is therefore to connect the different layers in the development process. We propose therefore moving away from purely textual specifications towards more visual specifications, connecting to formal UID specifications and connected to executable design models, for which code can be generated from those models. The proposed development process is shown in Figure 6.

4 Engineering methods

To current and proposed development processes can be characterized by the following engineering methods:

Engineering methods current process	Engineering methods proposed process
Create UID Manual created User Interaction Design specifications in Word (functional and non-functional requirements)	Create UID Derive user interaction designs (UID) by (1) scripting visual requirements simulator (functional requirements) and (2) rapid prototyping of executable reference architecture (non-functional requirements)
*	Formalize UID Create Domain Specific Language (DSL) to formalize the UID
Create traceability between left side of V-model Manual traceability matrix from UNS to UID to detailed design	Create traceability between left side of V-model Use the DSL for (automatic) traceability between UID, system design, detailed design and implementation
Create traceability between left and right side of V-model	Create traceability between left and right side of V-model Currently used tooling must be made interoperable: OSCL-compliant link interface between UNS, UID and verification and validation tooling
Requirements Validation System requirements specifications are verified and validated with working prototypes on real system	Requirements Validation System requirements specifications are verified and validated with software prototypes on a 3D workstation
Requirements Verification Requirements are developed in Caliber in a product family context, extracted to Microsoft Office (specific template) and then archived in Oracle Agile for formal reviewing and approval. This leads to System Designs created in Microsoft Office, extracted to Oracle Agile for formal reviewing and approval. Test designs and Test Cases are created in HP Application Lifecycle Management extracted to Oracle Agile for reviewing	Requirements Verification Requirements developed in Caliber in product family context, easily exportable in customizable templates to Microsoft Office, preferably able to archive automatically in Oracle Agile. Requirements are "live" available in HP ALM and direct traceability can be set and maintained in their Product Family context. From HP ALM direct exports in fully customizable templates to Microsoft Office should be possible to allow for easy archiving. Preferably able to archive automatically in Oracle Agile. (See chapter 5 for more details).

and formal approval. The actual test execution starts and evidence is added to HP ALM, after test execution manual extractions into Microsoft Office are created for formal reviewing and archiving in Oracle Agile.	
--	--

The results are quantified and validated based on industrial user scenarios. The validation includes:

- (Improve) level of interoperability between applications
 - o Validate if tooling is interoperable for complete traceability from requirements to verification.
- (Reduction) of development effort by improving of early feedback on extra-functional requirements.
 - o Validate if simulation tooling can gather user needs into a model instead of real prototyping.

Three user scenarios are defined in the Philips Healthcare domain according to the metrics defined before. Notice that these serve as guides for defining and developing the model and tool chain.

4.1 Scenario 1: Orientation of x-ray image on monitor

4.1.1 User Needs

The x-ray image on the monitor represents a two-dimensionaal image of the patient on the table.

In case of a diagnostic examination, the image needs to be presented as: head up; patient left on right side monitor, independent upon the actual position/orientation of the patient with respect to the x-ray table.

However, in case of some interventional examinations, objects in the patient (e.g. needles) have to be manipulated using the x-ray image. To improve hand-eye coordination, a different image orientation on the monitor may be required.

4.1.2 Case study description

The orientation of the x-ray image on monitor is affected by a lot of variables:

- patient orientation on x-ray table (feet to left or right; lying on back, belly, left side or right side)
- orientation of x-ray beam with respect to table
- orientation of detector with respect to x-ray beam
- image processing (image rotation, left-right swap)

The required image orientation depends upon the particular examination and the physian using the system (radiologist, cardiologist, surgeon). Because of the large set of variables, visualisation tooling is required to explore the real user needs and to find the optimal requirements.

4.2 Scenario 2: Setting x-ray beam projection with a joystick

4.2.1 User needs:

To visualize anatomical structures in the patient and to minimize overlap of structures, an angled x-ray beam projection is required. Via a joystick on the user interface module the operator can set the angulation and rotation angle of the x-ray beam with respect to the x-ray table and patient. In current systems, one-to-one relations exist between image on monitor, x-ray beam projection, detector movements and physical movement axes. In new systems, the one-to-one relations are not present anymore.

4.2.2 case study description:

To find the optimal requirements for the control of x-ray beam projection via a joystick, various models for the hand-eye coordination need to be explored. Possible references for the operator are:

- image on monitor
- virtual movement of x-ray beam
- physical movement of x-ray detector
- physical movement axes.

Because of the large set of variables, visualisation tooling is required to explore the various models for the hand-eye coordination.

4.3 Scenario 3: Movement direction of bolus chase

4.3.1 user needs:

To visualize obstructions in the blood vessels in the legs, a so called bolus chase technique is used. At the start of the bolus chase a contrast medium (bolus) is injected in the lower part of the aorta. Together with the blood, the contrast medium flows towards the toes. Because the x-ray beam cannot cover the complete area from injection point to the toes, the x-ray beam is moved towards the toes (or the toes are moved towards the x-ray beam). The movement speed is controlled by the operator using the image on the monitor.

4.3.2 case study description:

Adding more table configurations to the Allura system resulted in various implementations for bolus chase movement directions. The following configurations are taken into account:

- standard x-ray table (AD5, AD7)
- OR table with universal table top
- OR table with reversed table top
- ceiling stand with/without X-Y movement

Feedback from the field has shown that implemented movement directions were not optimal.

Can visualisation tooling help in finding the correct requirements for the bolus chase movement direction?

5 Engineering methods Detailed descriptions

5.1.1 Requirements verification

Philips defined several layers in the V-model. Validation, Verification, Development. This engineering method focusses on Verification. Validation focusses on asking the customer if the product is created according to their needs, more high level orientation (e.g. the system must be safe to use).

In short Verification means collecting evidence that our system performed as specified in a specific requirement this is a more detailed level than validation (e.g. The system's power on sequence may take a maximum of 360 seconds).

Verification is done on a (nearly) complete system to allow for tests that are similar to hospital devices. It is easily comparable to manufacturing a car; at some point crash tests need to be executed to collect the evidence the car is safe in extreme conditions. These crash tests need to be done with a complete car to simulate conditions that reflect consumer use. Philips Healthcare does the same with its X-ray systems, running reliability tests and performance tests to guarantee quality (meeting specified requirements) on a full system.

Current use (see also Annex III for high level visual representation of the engineering method)

Pre-Condition	Engineering Activity as Steps	Post-Condition
1a. Authorized list of requirements (requirements archive - Caliber) 1b. Authorized Project Assignment 1c. Authorized Master Test and Release Plan 2. Authorized System Design Specification 3a. Requirement implemented (SW archive - ClearCase) 3b. Requirement integrated (executable on testsystem) 3c. Testsystem ready for test	1. Create TestDesign based on precondition 1a in QualityCenter. - Review 2. Create TestScript based on TestDesign and precondition 1, 2 in QC. - Review 3. Add traceability between precondition 1, TestDesigns and TestScripts (manual with Excel link : Caliber <-> QC) - Review 4. Execute TestScript (QC) on testbase (testsystems) - Manual test execution on testbase, logged in QC, defects logged in (ClearQuest), precondition 3 needed. 5. Generate TestTraceabilityReport from QualityCenter - Review 6. Generate TestVerificationReport from QualityCenter to Word 7. Add TVR to Agile (PLM) pdf file - Review & Authorization	Authorized verification report containing the verified requirement(s)

Wish for the future (conceptual discription):

Pre-Condition	Engineering Activity as Steps	Post-Condition
Applications that can share data in its context	<p>1. Create requirements in an application for Requirements Lifecycle Management and make a requirements document available in the Application for managing documentation lifecycles (PLDM).</p> <p>1a. The OSLC interface enables the requirement with its content and in its context (traceability between requirements) as soon as it is in an approved state.</p> <p>2. In the Test Management & Execution application the requirement can directly be seen with its content and in its context (traceability between requirements).</p> <p>3a. A report of the requirements can be created in both the RLCM software or the test software into the same template for reviewing.</p> <p>3b. <i>A System Design Specification (SDS) is created and available in the PLDM.</i></p> <p>4. A Test Design is created with the Requirements and SDS as input and traceability is set.</p> <p>4a. As soon as the Requirement changes this leads to alerts / triggers in all linked applications.</p> <p>4b. From the RLCM software perspective the traceability can be seen with its test designs and test case context.</p> <p>5. As soon as all Test Designs and Test cases are created, different reports can be generated, traceability matrix, test design overview, test case overview etc. and directly available in PLDM.</p> <p>6. As soon as approvals in the PLDM application are given triggers are visible in the linked applications.</p> <p>6a. In the Test management & execution software the test designs and cases are set on status reviewed / approved.</p> <p>7. Test Execution can be started and as soon as a test case is executed the status of that test case is updated to all linked applications (live status updates).</p> <p>7a. in the requirements management software the requirement with its linked test case and including the status is visible and can be reported on.</p>	Authorized verification report containing the verified requirement(s) without manual push and pull interfaces and extra manual checks on data integrity and consistency.



6 Terms, Abbreviations and Definitions

n.a.	

Table 6-1: Terms, Abbreviations and Definitions

7 References

[Author, Year]	Authors; <i>Title</i> ; Publication data (document reference)
n.a.	n.a.



8 Annex I: Detailed Descriptions of the Engineering Methods

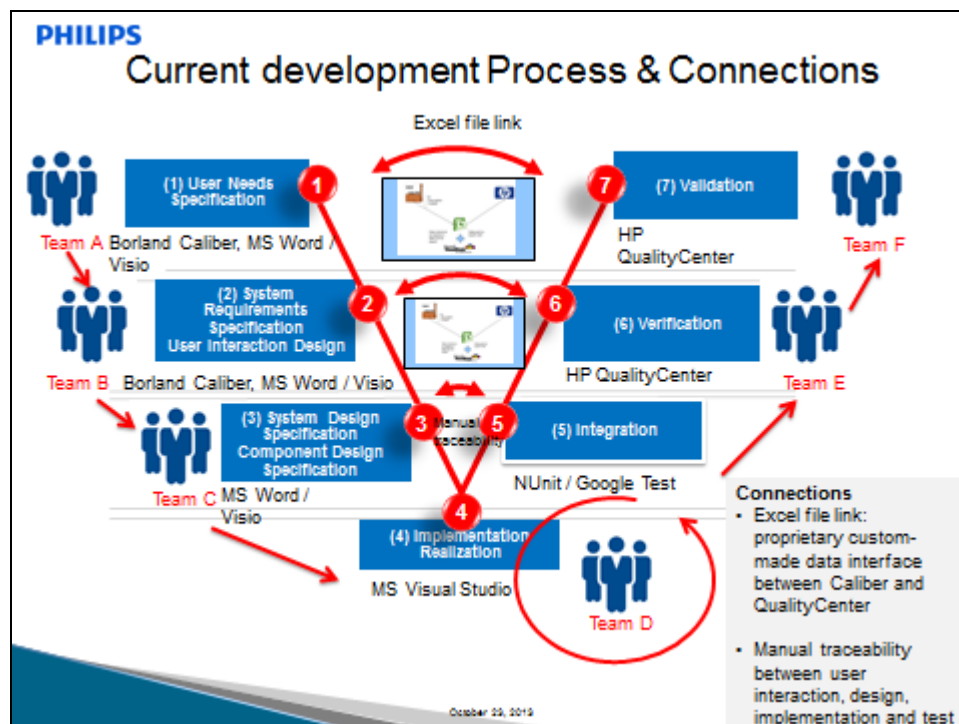
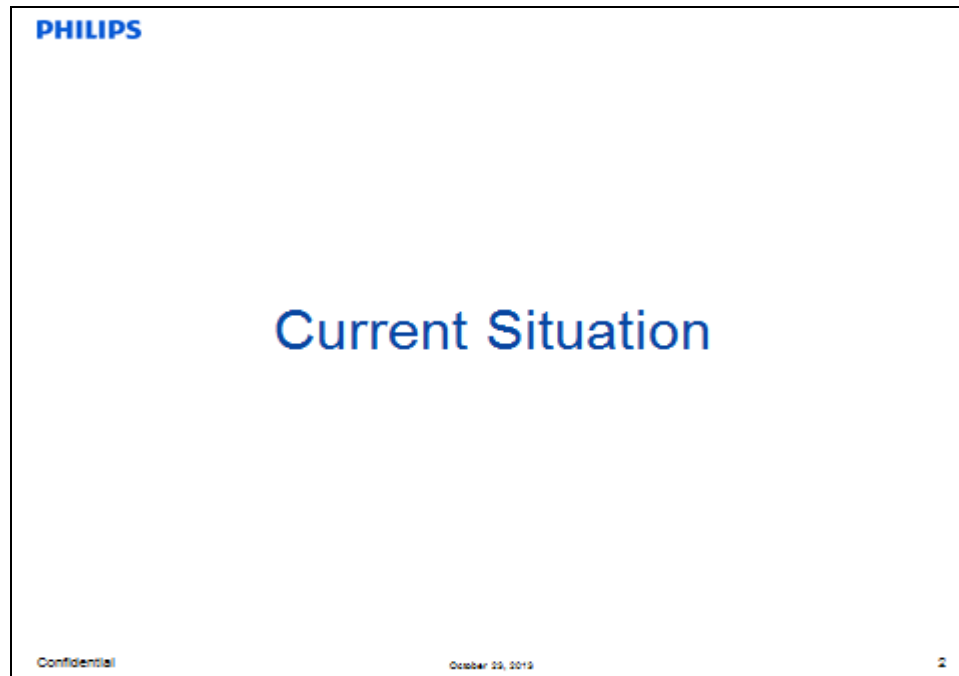
See chapter 5

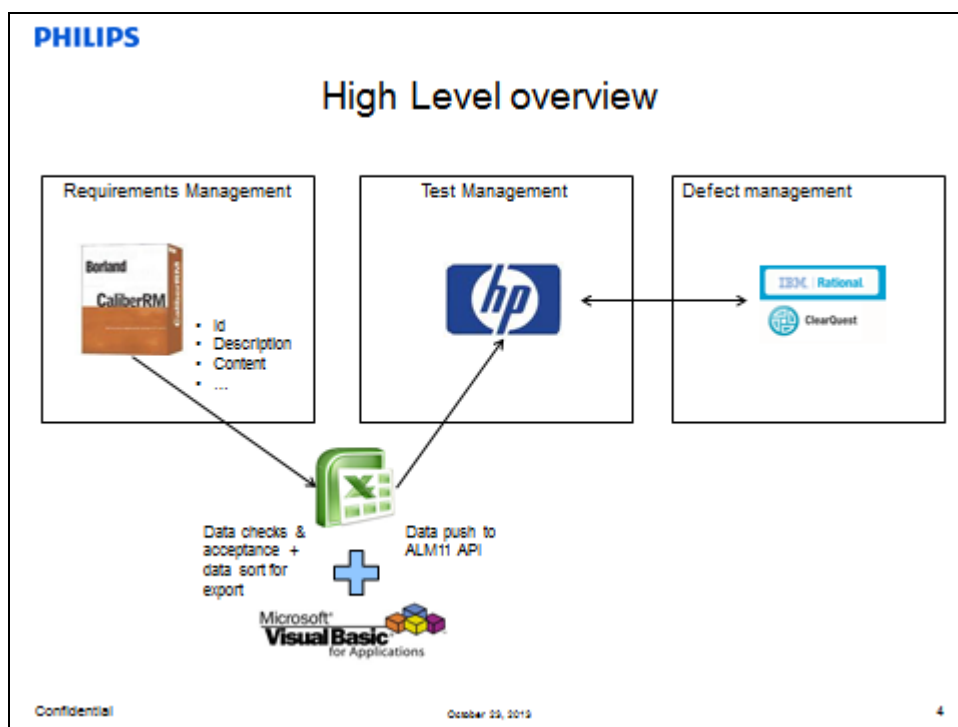
9 Annex II: Technology Base Line & Progress Beyond

This information will be collected globally, and the respective part will be inserted here. Basically it could be something like a table with a row for each engineering method and a column for the current functionality, which is the technology baseline (e.g., “data has to be transferred by hand”), and a column for the expected progress in CRYSTAL (e.g., to be implemented in CRYSTAL / “future work”).

The exact content of this section will be defined in the next technical Board Meeting.

10 Annex III: Visual representation of current situation

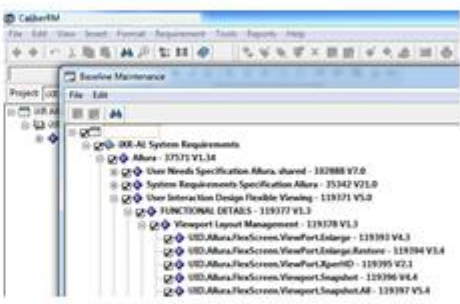




PHILIPS

Caliber

- Define requirements (name, description)
- Add meta data
 - Scope
 - Status
 - Priority
 - Owner
 - Traceability
- Create baseline of requirements set



The screenshot shows the CaliberRM application window. The 'Project' pane on the left displays a tree structure of requirements and snapshots. The main window shows a list of requirements, including 'User Needs Specification', 'System Requirements Specification', and 'Functional Details'. Each requirement is associated with a specific version (e.g., V1.0, V1.1, V1.2, V1.3, V1.4).

Confidential

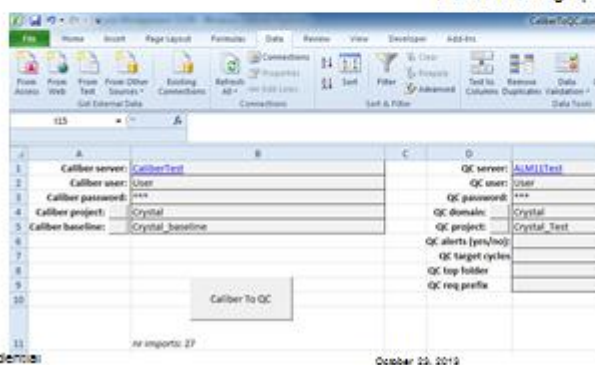
October 22, 2012

5

PHILIPS

Excel

- Select correct Caliber server
- Set Caliber user
- Set password
- Select Caliber project
- Select baseline



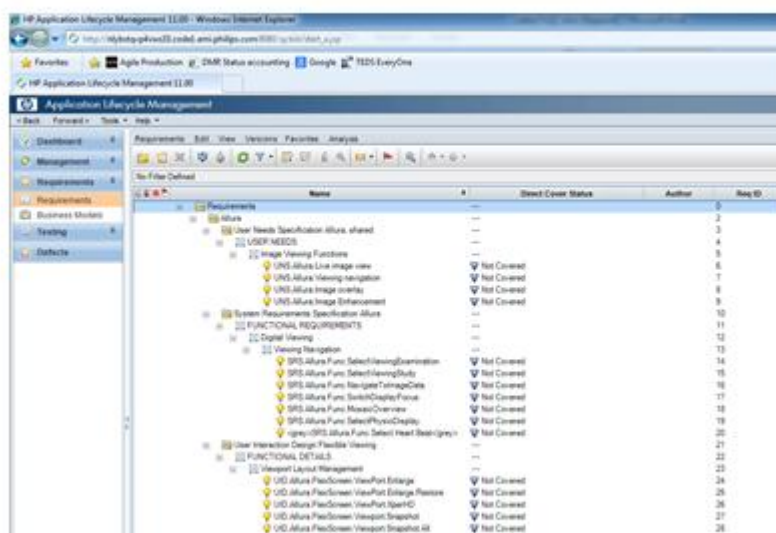
Confidential

© 2004 Blackwell Publishing Ltd *Journal of Internal Medicine* 255: 105–112

6

PHILIPS

ALM11



Confidential

December 20, 2008

7

PHILIPS

Wishes for the future

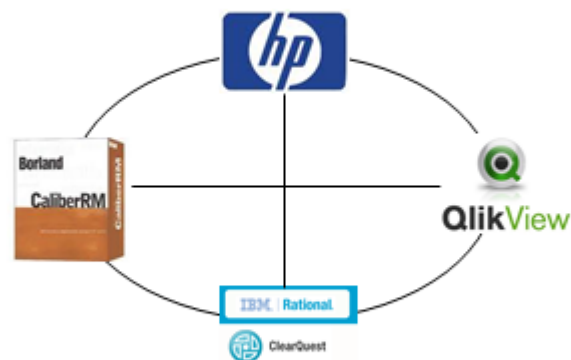
Confidential

October 23, 2013

8

PHILIPS

High level overview phase 1– basic OSLC



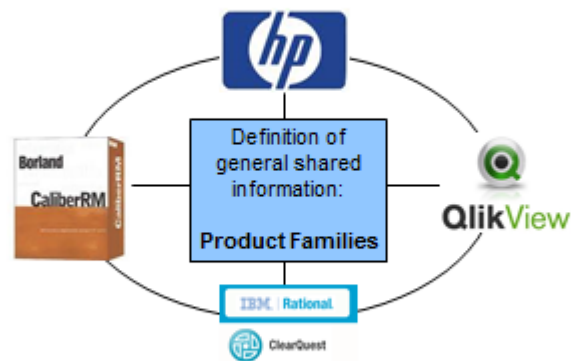
Confidential

October 23, 2013

9

PHILIPS

High level overview phase 2 – Smart OSLC



Push & Pull focus from application A to B for general information

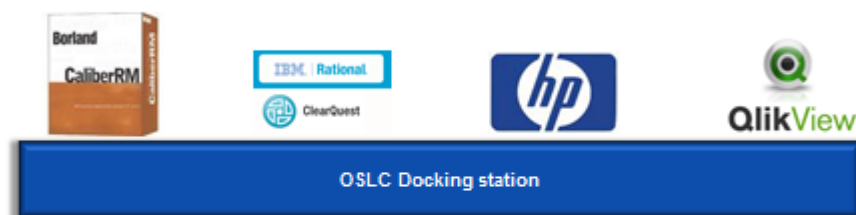
Confidential

October 23, 2013

10

PHILIPS

High level overview phase 3 – OSLC docking station



- Fully shareable content
- Interoperability between applications
- One industrial standard
- Generic definition of input & output possible
- Maintainable field mapping with profiles
- Push & Pull

Confidential

October 23, 2013

11