 THE RESEARCH LEADING TO THESE RESULTS HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S SEVENTH FRAMEWORK PROGRAM (FP7/2007-2013) FOR CRYSTAL – CRITICAL SYSTEM ENGINEERING ACCELERATION JOINT UNDERTAKING UNDER GRANT AGREEMENT N° 332830 AND FROM SPECIFIC NATIONAL PROGRAMS AND / OR FUNDING AUTHORITIES.

CRitical S**YST**em Engineering **A**cce**L**eration

Use Case Development Report UC401
Medical procedures in an interventional X-ray system

D401.901

DOCUMENT INFORMATION

Project	CRYSTAL
Grant Agreement No.	ARTEMIS-2012-1-332830
Deliverable Title	Use Case Development Report UC401 Medical procedures in an Interventional X-ray system
Deliverable No.	D401.901
Dissemination Level	CO
Nature	R
Document Version	V1.0
Date	2014-04-30
Filename	CRYSTAL_D_401_901_v1.0.doc
Contact	R. Albers
Organization	Philips
Phone	+31 6 10084377
E-Mail	r.albers@philips.com

AUTHORS TABLE

Name	Company	E-Mail
Jan Joosten	Philips	Jan.h.joosten@philips.com
R. Albers	Philips	r.albers@philips.com
T. de Laet	Philips	Thomas.de.Laet@philips.com
J. Hooman	TNO	jozef.hooman@tno.nl
T. van Velzen	IBM NL	tonv@nl.ibm.com
C. Bratosin	TNO	carmen.bratosin@tno.nl

CHANGE HISTORY

Version	Date	Reason for Change	Pages Affected
V1.00	2014-04-29	Approved version	

CONTENT

Content of Figures

Content of Tables

1 Introduction

1.1 Role of this document

The intention of this Use Case Development Report is to provide an (annual) overview on the status of engineering methods, engineering environment and improvement activities related to the development of Medical Procedures in an Interventional X-ray system. For more information see High level description of use case and context. As depicted in the figure below, its content will vary over time, in line with the phase of the Crystal project it is reporting upon.

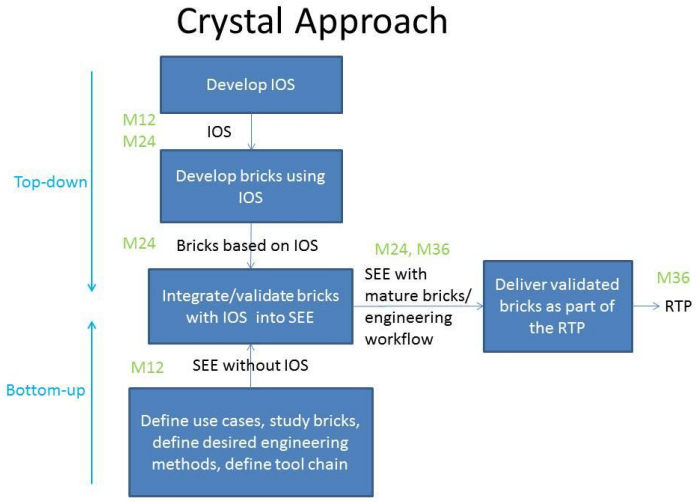


Figure 1: the content of this document will vary over time, in line with the phase of the Crystal project it is reporting upon.

1.2 Relationship to other CRYSTAL Documents

The figure below provides a general overview of the internal structure of the Crystal project. This work package is part of the Healthcare domain (SP4). Its information and reports are input for WP6.

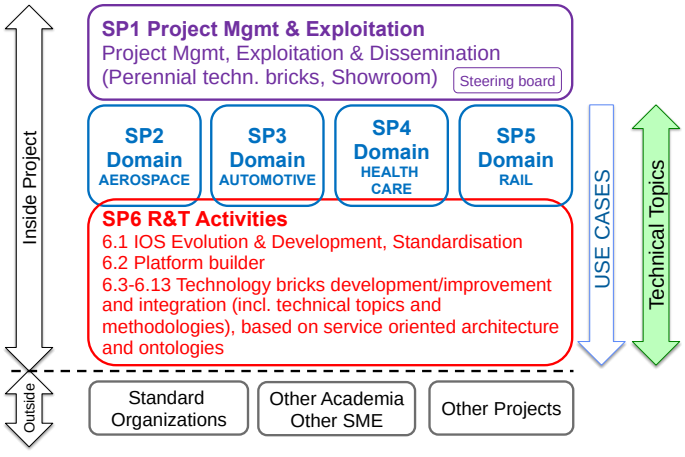


Figure 2: Crystal project structure

This document is closely related to the Use Case Definition Report for Medical Procedures in an Interventional X-ray system. Where the *Use Case Definition Report* elaborates on the technical details and the decision making process, the *Use Case Development Report* is used to provide a condensed overview of the planned and scheduled improvement activities, with an Executive summary on the description of work and its conclusions.

1.3 Structure of this document

This document describes the improvements in the development process for UseCase 4.1 achieved in the context the Crystal project. More detailed information on Use Case 4.1 is available in [ref1] and in High level description of use case and context.

This document starts with a description of the current development process for the positioning system of theInterventional X-ray system[1] and the challenges at the start of the Crystal activities for WP4.1 (M0).

Based on the Crystal timeline M0 – M12 and M12 –M36, this document reports the status of the Crystal activities by means of three views:

- ***Engineering Workflow view***: engineering workflow with improvement steps (chapter ..)
- ***System Engineering Environment view***: tools with interfaces, improvement steps (chapter ..)
- ***Modelling view***, concerning specifications and models (chapter ..)

For each of these views this report will give:

- the status at M0
- the results of the Crystal activities with respect to Engineering Workflow, Modelling and Engineering Environment
- The status at M12
- Planned activities in Crystal context for the period M12 – M24 – M36.

[1] The positioning system part of an X-ray system consists of an adjustable table to support the patient, an adjustable stand with X-ray tube and detector and a user interface to enable the doctor to control the position of patient, X-ray tube and detector. Read more in section 2.2.

2 Use Case 4.1 Medical Procedures in an Interventional X-ray System

This paragraph provides a high-level presentation of the use case, the product developed, and the problem space as perceived from the end user perspective. It provides a refinement of the use case concerning a robotic positioning system and it lists the requirements concerning interoperable tooling.
More detailed information on Use Case 4.1 is available in [ref1] and in High level description of use case and context.

2.1 Introduction

The use cases of Philips Healthcare concern the mechatronics control part of an interventional X-ray system. It is important to be able to incorporate medical innovations quickly in such systems. This is challenging, because high quality standards have to be met. To meet these goals, Philips investigates a migration towards a new ***component-based architecture*** and an improved ***model-based iterative development process***, supported by ***interoperable tooling***. The three use cases concern different layers of control of an interventional X-ray system.

The development is used to investigate interoperable tooling for:

- Component-based development
- Multi-disciplinary modeling and simulation, supporting continuous integration
- Code generation from models
- Real-time behaviour and performance analysis
- Test and integration

The primary objective of this Use Case is to achieve the desired speed profiles and positions of the robotic positioning system. The Use Case is focusing on mixed physics/data-based modelling and simulation of sensorial and mechanical uncertainties in robotic positioning systems and on how these uncertainties translate to the performance of the systems in their environment and therefore on the safety of the complete system.

2.2 Medical context

Image-guided interventions and therapy demand for an eased workflow with regards to manoeuvring patient examination table and stands. The integration with various components into the OR and Cathlab makes safe positioning of the X-ray system challenging. As an example, see the figure below where a Hybrid OR room is shown, full of equipment.

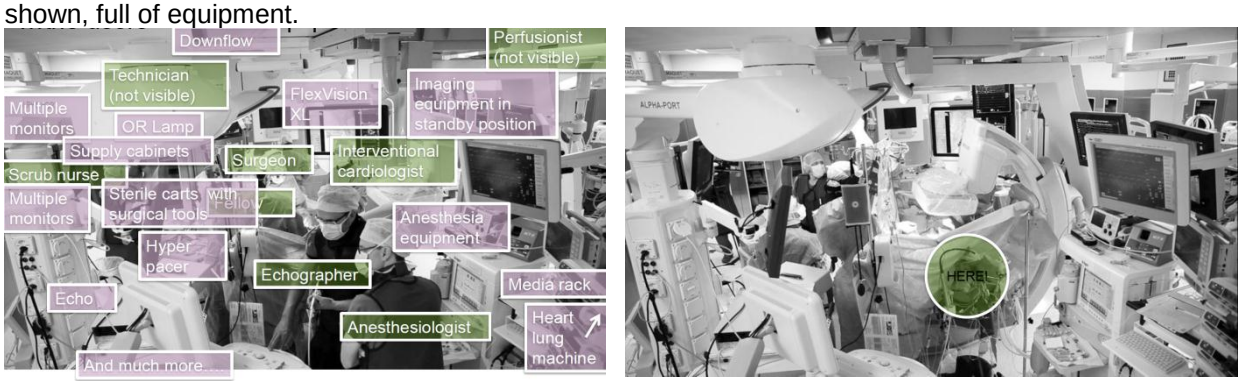


Figure 3: Hybrid OR with (left) all equipment and (right) the position of the patient in the room.

Ideally interventional X-ray camera's would be small and light, enabling easy control, not restricting in anyway the doctor in doing clinical procedures. Unfortunately this is not the case; in real life we have a heavy camera (consisting of tube, collimator and flat detector, etc.) which needs heavy and large mechanics. Moreover we need a large patient examination table to support the desired position of our patient.

The mechanical degrees of freedom

The mechanics part of the X-ray system contains a patient examination table to support the patient and a stand that carries the X-ray tube and detector. Its user interfaces allow the clinical staff to move and position the patient and the X-ray beam in order to generate images or series of images from medically appropriate projections.

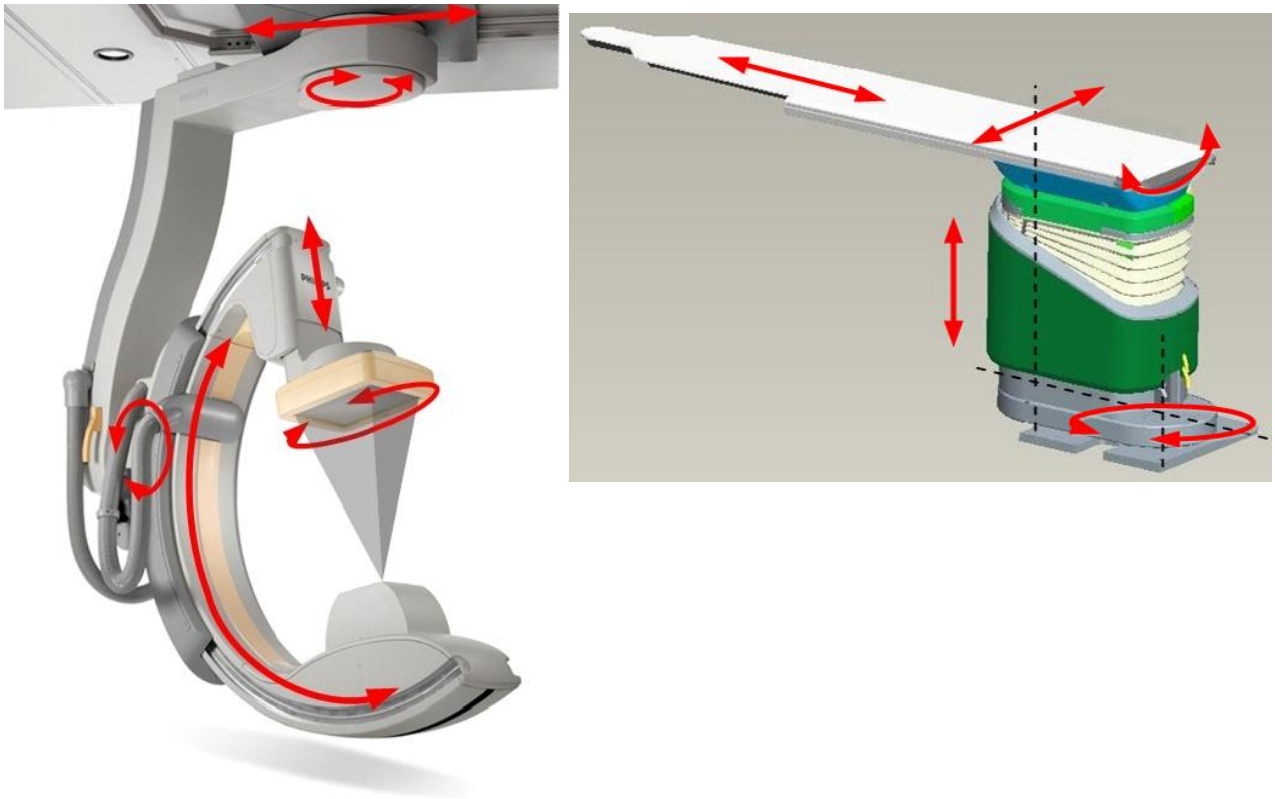


Figure 4: A sample product configuration and its axis of movement; ceiling suspended stand and patient support table.

Several challenges appear at the horizon when designing a system where there is no one-to-one relation between user interface buttons/joysticks and the basic movement axes:

(1) Limitations of patient oriented movements by hardware restrictions. Philips wants to gain insight in and demonstrate the possibilities and limitations of the system to applicants or stake holders.

(2) Multiple scenarios where detector or tube might collide with the patient examination table stand. The physical movement ranges of all individual axes are not limited such that no collisions can occur. In order to prevent collisions path guarding software checks for impending collisions. This way a certain clearing distance is taken into account. From an end users point of view the clearing distance should be as small as possible in order to get an optimal view on a patient. From machine point of view the clearing distance should be as large as possible in order to guarantee NO collisions in any circumstance. Due to uncertainty of all individual axes positions there is uncertainty of the position where a collision might occur. Philips want to gain insight in to determine how the smallest possible clearing distance between C-arc (with collimator and detector) and patient examination table stand can be achieved given axes accuracies in multiple scenarios.

2.3 Challenges at M0

Various trends and drivers challenge the current engineering workflow, including:

- Increased design complexity due to higher demands on flexibility in the clinical room layout, clinical requests for patient oriented movements, and increased integration with other medical equipment.
- Increased variability triggered by efforts to adapt the same product platform for a broader audience (e.g. in intended use, value segment, or notified bodies)
- Reduce time to market.

While the current way of working is rather document driven and code-centric, various opportunities for improvement in the engineering workflow are identified:

- Information is stored at various places and thus hard to find.
- Information is typically replicated in the various tools used and thus hard to keep consistent.
- Requirements are predominantly natural language based which may not be the most effective means of communication, makes it hard to determine quality levels, and hampers re-use throughout the engineering workflow. Frequently ambiguous, error-prone, or misinterpreted.
- Models aren't re-used throughout the engineering workflow, thus it is hard to preserve the overall consistency in requirements, design, implementation, and verification/validation.
- Model engineering is done manually while tools can assist in establishing a level of abstraction.
- Maintaining the traceability of information is a manual exercise which is hard to keep consistent across the various tool environments.

Models are recognized as a means to counter complexity by raising the level of abstraction:

- ✓ As requirements aid by defining the desired product behaviour (e.g. behaviour models)
- ✓ As design aid by defining the actual product behaviour (e.g. architectural / structural models)
- ✓ As implementation aid via code generation.
- ✓ As verification aid by predicting product behaviour (e.g. emulation or simulation models)
- ✓ As validation aid by providing early clinical feedback on the product behaviour.

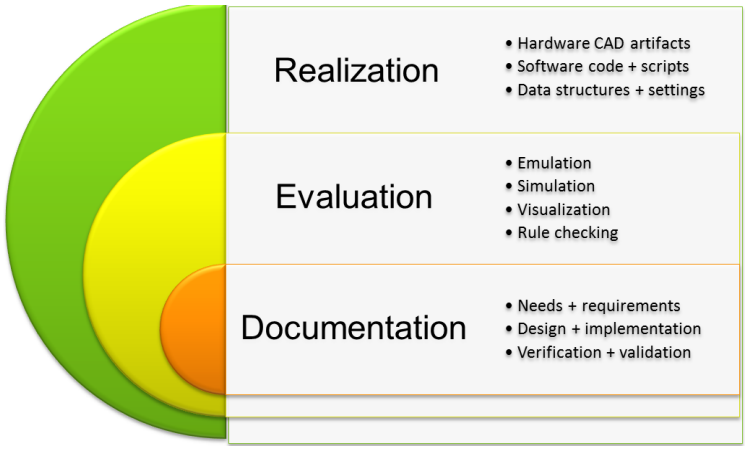


Figure 5: Evolutionary path towards model re-use throughout the engineering workflow

Re-use of models throughout the engineering workflow is perceived as a strategic and breakthrough systems engineering innovation. While it aligns closely to most of the challenges and positively affects the concerns on information scattering, replication, consistency, quality levels, re-use, ambiguity, or traceability, it doesn't come for free.

This breakthrough innovation challenges the organizations capabilities, especially in the area of model engineering, model-based development, and model-based simulation and verification and poses new requirements towards its processes, tools, and engineers. Other factors to consider while re-using models are the potential impact on product safety (as depicted in Figure 5 above) or the chance for detecting or the cost for resolving model deficiencies.

Within the current time frame, the Crystal activities were split along the following themes;

a) *Individual models and the ecosystem architecture.*
The activities at this level are primarily concerned with the content of an individual model, and its associated quality attributes as defined under ISO 25010. Its ultimate goal; develop an ecosystem of models that together cover all critical to satisfaction/quality aspects of a product family. The models ability for re-use throughout the engineering workflow is another example of a factor to take into consideration while defining the scope and context of individual models.

b) *Model engineering infrastructure.*
Activities related to the simulation and visualization architecture and its underlying components, enabling the extraction, distribution, and processing of data that facilitates in the development and verification of model content. The tools in this category may exhibit product specific behaviour.

c) *Optimizing the engineering workflow.*
Emphasis here is on the interoperability between tools and the smooth exchange of engineering artefacts and associated Meta-data while heading towards an environment that supports continues build and integration. The tools are typically generic in nature and do not exhibit product specific behaviour.

d) *Institutionalizing changes to the Systems Engineering Environment*
Change management activities required for embedding alternative tools, technologies, or processes in the standing R&D organization, thus ensuring on its sustained adaptation within the domain or industry. It includes activities required for technology demonstrators and such.

3 Engineering Workflow

This paragraph describes the development activities related to the engineering workflow for this work package. It describes the initiatives started, and the envisioned engineering workflow, planned to be available at the M36 milestone. It highlights the engineering methods associated with this work package and concludes this section with a list of artefacts relevant for this work package.

3.1 Engineering workflow at M0

WP4.1 targets improvements that focus on the part of the V-model indicated in Figure 6; it zooms in on this focus; it highlights the engineering workflow as was the current way of working prior to Crystal.

Use Case 4.1 Medical Procedures in an Interventional X-ray System concerns the higher levels in the X-ray architecture (UI design, overall architecture, supervision functions) and the higher levels in the development process as depicted in the following workflow diagram.

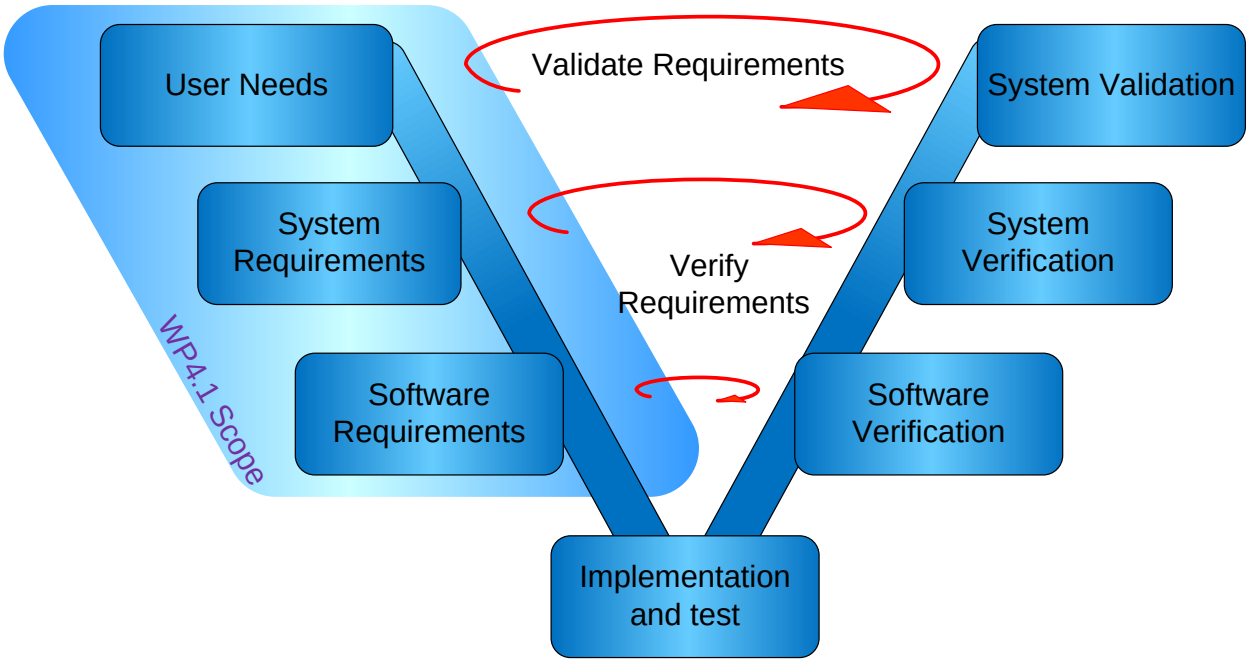


Figure 6: the WP4.1 scope: the higher levels in the development process.

Characteristics of the engineering workflow at M0:

- Because they are subject to strict regulations, healthcare systems are developed in a well-defined development process, currently following the traditional V-model.
- Advantages: well documented record and audit-trail of process and product. Natural fit to engineers way of working.
- Disadvantages: late system integration, extensive documentation.

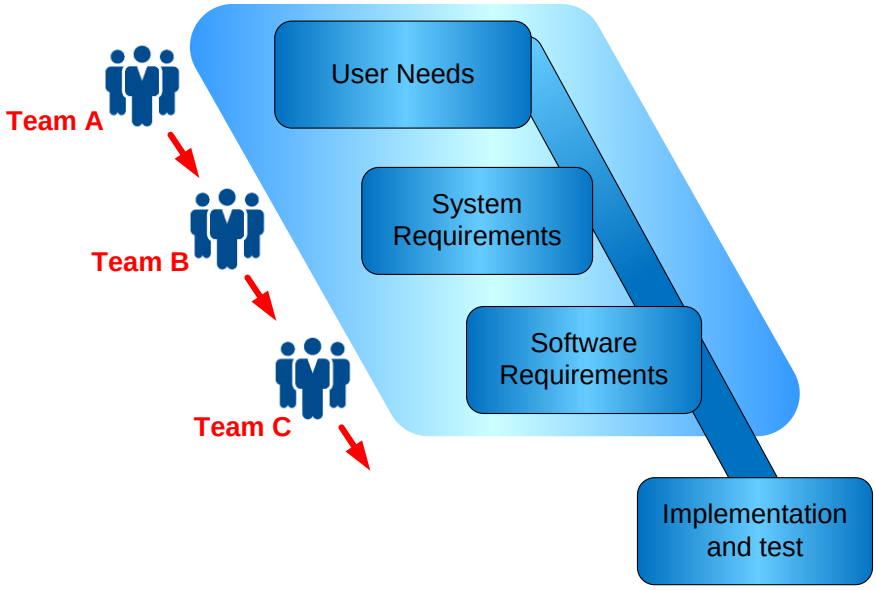


Figure 7: Incremental way of working with one multidisciplinary team covering a large part of the development process.

A number of trends[2] forced to a more parallel approach of developing X-ray systems:

- Growing complexity of the systems and with it the need for integration of subsystems into a system.
- With the introduction of software the whole approach needed redefinition, agile way of working, multiple software teams handling the increasing demand of software features.
- The last complexity factor is today's market where shorter time to market is needed to speed up innovation and handle price reducing demands. This all translates into variants, product families and configurations and configuration management.

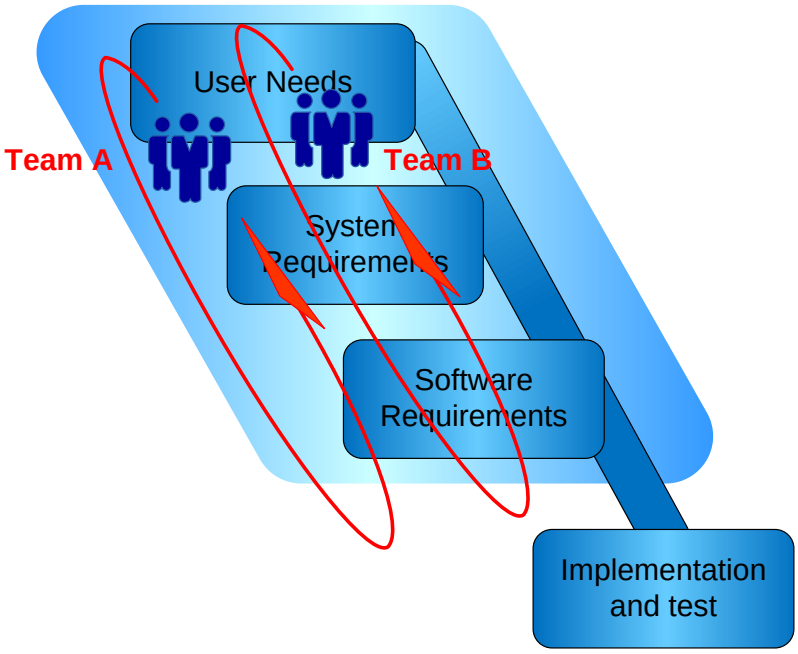


Figure 8: Parallel development approach.

[2] See also section 2.3 "Challenges at M0".

3.2 Initiatives started

The table below provides an overview of the previously mentioned themes and the activities initiated in the M12 time frame.

	Activity	Individual models	Modelling infrastructure	Workflow optimization	Change management
A1	Early concept validation of mechatronics using 3D virtual reality viewer	■		■	
A2	Early visual verification of system requirements using 2D viewer	■		■	
A3	Functional Requirements Analyzing and Formalization using DSL	■		■	
A4	Infrastructure to early visual verification visualize using 3D virtual reality viewer		■		
A5	Early visual verification of formal requirements in DSL using 3D viewer	■	■	■	
A6	Couple DSL to requirements management tooling using OSLC		■		
A7	Early verification of system design concepts using 3D viewer	■		■	
A8	Early verification of system design concepts using demonstrator	■			
A9	Early verification of mechatronics design concepts using Matlab	■			
A10	Early verification of mechatronics design concepts using Matlab and 3D viewer	■		■	
A11	Early verification of mechatronics design concepts using demonstrator	■			
A12	Early verification of software design concepts using POOSL	■		■	
A13	Early verification of software design concepts using demonstrator	■			
A14	Coupling requirements to verification test cases using HPQC		■		■
A15	M9 Demonstrator Caliber – HPQC – IBM RQM				■
A16	M12 Demonstrator: Integrated demo WP4.1 + WP4.3				■

Table 1 Main themes in relation to activities reported for WP4.1

Figure 9 depicts how the 16 activities of M12 4.1 map on general development process in a critical system engineering environment.

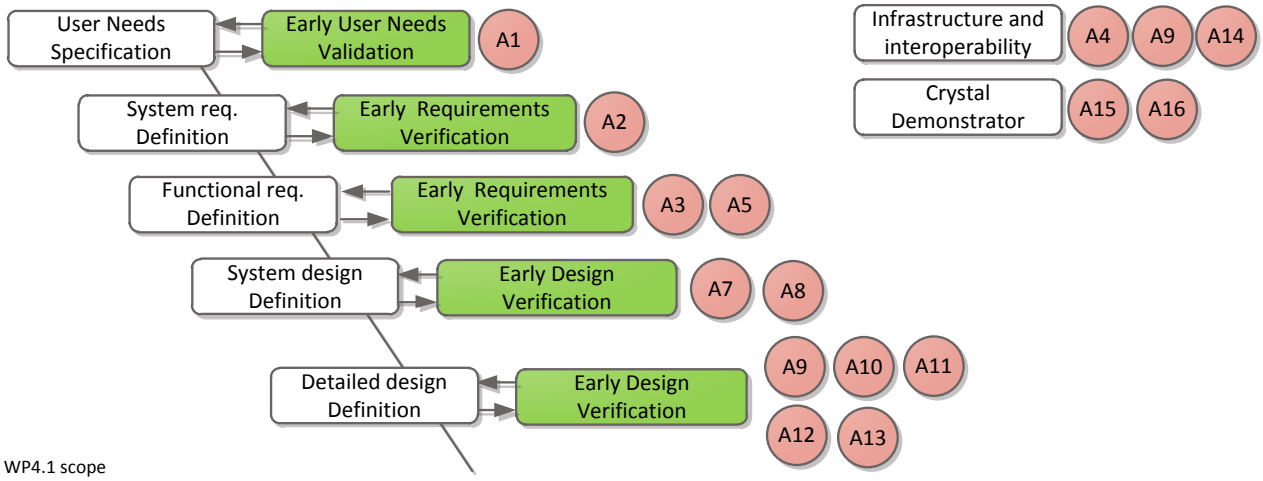


Figure 9: Mapping of activities on development process

3.2.1 Activity A1: Early concept validation of mechatronics using 3D virtual reality viewer

For more information about frontal stand and patient examination table see section 2.2.

The reason for the new technology

Full understanding of user needs is required to avoid mistakes and corrections in the project and – if misunderstandings are not detected – a non-optimal product. It is crucial for the acceptance of a product in the field.

The most efficient way to get full understanding of user needs is to visualize the application situation, let the user explain his needs, get early feedback on new concepts. Obviously physical models can be used for this goal, but the use of 3D simulations is faster and more flexible.

The stakeholders of the new technology

Stakeholders for the new technology are medical end users, (clinical) marketers, system-, software- and mechanical designers.

Approach

To enable a discussion about all medical application aspects, the 3D interactive visualization covers not only the X-ray system but also its complete environment: ceiling suspensions with displays, anaesthetic instruments, control room, lighting, etc. In this way also plans for a future examination room can be visualised.

To analyze the application situation in detail, the observer can place a virtual 3D camera at any place in the examination room and zoom in to any detail. Doctors and assistants can be placed at any place in the room to investigate the movement freedom. A standard UI is available to move patient support table and stand.

The 3D visualization allows the stakeholders to discuss and analyze critical application scenarios, while the tool visualizes all relevant aspects of the system and room under study. In this way of obtaining early feedback on the system design is also called pre-validation or concept validation.

Activities during last year

- Realization of the the 3D interactive visualization tool;
- Incorporation of various tool options, system and room configurations;
- Exercises with various medical use cases (e.g. system start, system stand by, different medical procedures).
- Discussions with medical users based on this 3D visualization (in house, on exibitions);

Results

- Tool with extensions realized;
- Extensive knowledge of modeling X-ray systems and exam rooms in Virtual Reality Tool (Vizard);
- Knowledge was gained about usability in various medical procedures

Tool requirements

- 3D interactive visualization must be compatible with internal and external (third party) visualization tools
- One common interactive visualization tool for all sales, application and engineering activities (‘E2E tool: usable for sales and development (from user needs to design/implementation)).
- X-ray systems and their environment shall be modeled.

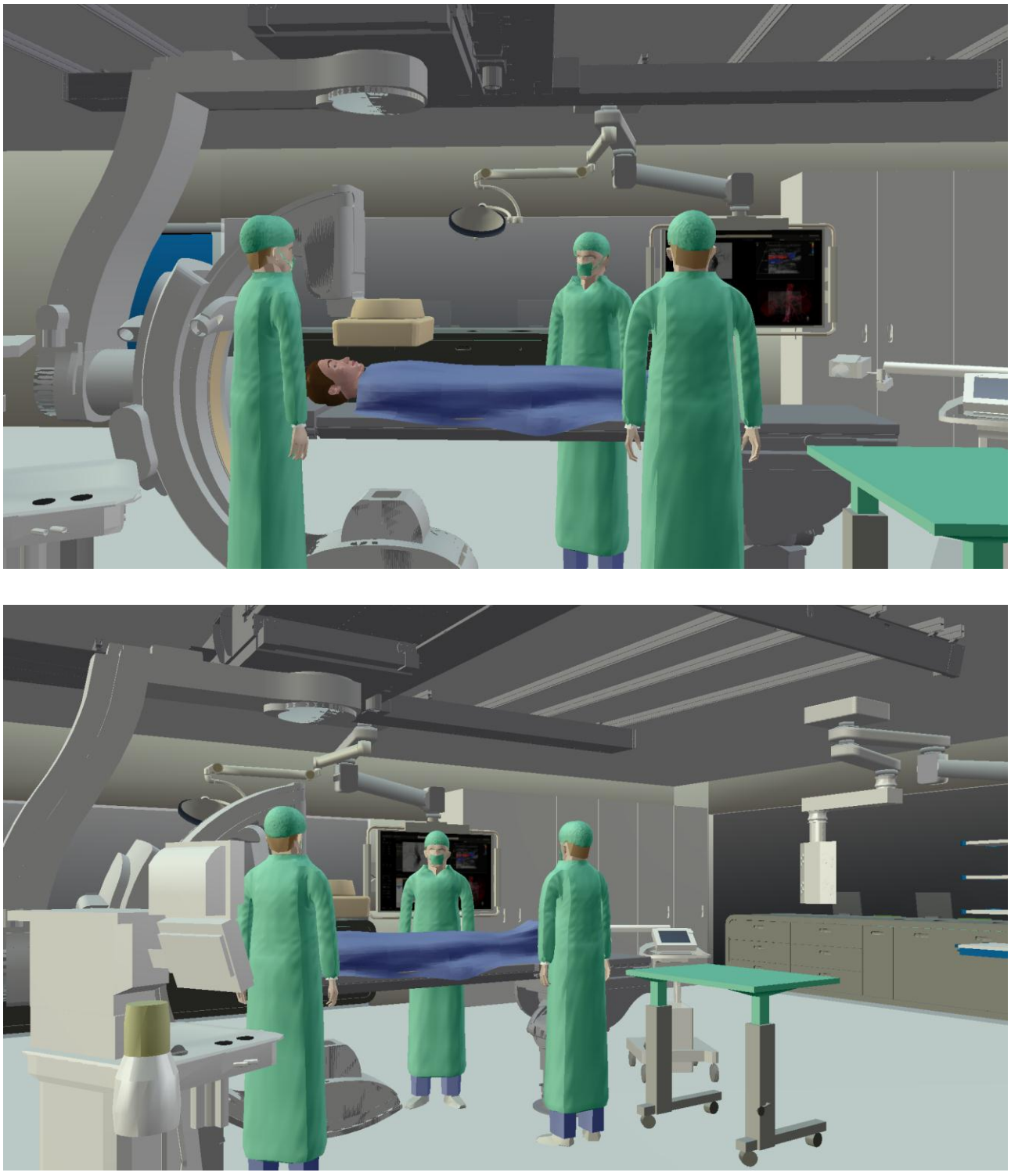


Figure 10:Two screenshots of the output of the 3D interactive visualization; the images not only cover the X-ray system but also its complete environment: ceiling suspensions with displays, anaesthetic instruments, control room, lighting, etc.

Follow up (M24 / M36)

Improvement of interchangeability of models

The 3D visualization tool is a dedicated standalone tool. It has been based on models of X-ray system and examination room (3D studio) that are not interchangeable between the visualization and simulation tools within the company. Coupling a ‘behavior model’ in Xposer to a 3D interactive visualization is a first step into an integrated Visualization Environment.

3.2.2 Activity A2: Early visual verification of system requirements using 2D viewer

For more information about frontal stand and patient examination table see section 2.2.

The reason for the new technology

Mechanical limitations in components may lead to restrictions in usability: limitations in patient accessibility for the various medical users (e.g. surgeon and anaesthetist), restrictions in patient coverage or limitations in access to (anaesthesia) instruments.

With 'classical' methods (CAD-drawings, spread sheets, manual drawings, descriptions) it is nearly impossible to get a good understanding of the consequences of mechanical design choices in the patient support table and the stand for the medical application ranges (movement ranges, patient scan ranges). This is especially difficult because of today's trends in X-ray systems: growing number of applications, growing number of configurations and a high pressure on development time. The combination of these trends asks for a method and tool that allows for an early and complete analysis to give insight in the consequences of design choices.

The 2D interactive visualization tool is mostly used in the overall design phase of a project.

It helps to:

- Determine whether choices made in the design of components restrict medical applications. If they do, how to solve the restrictions;
- Find the balance between usability and safety;
- Find the balance between usability and cost-price.

Examples: verifying the patient coverage for peripheral angiography, testing critical projections for cardiac applications, visualising accessibility of the patient for various medical users.

The stakeholders of the new technology

The primarily concerned parties for this visualization are medical application specialists, system designers and mechanical designers.

Approach

The 2D interactive visualization is a simulation tool that displays a 2D (top view) on the table and stand of the X-ray system. The display reflects the real X-ray system, with real configuration of components and real dimensions. A simulated user interface for the positioning system enables the user to control all the axis of the system and to verify the application ranges and patient access in an interactive way. Although it does not show the real time behavior, it simulates dynamic behavior (movements) of stand and patient support table.

Activities during last year

- Realization of the 2D visualization tool;
- Modeling and introduction of various configurations;
- Elaboration of various use cases (workshops in cooperation with application specialists and designers).

Results

The activities of the last year offered the following results

- 2D visualization tool realized; conclusion: 2D easier to create (less effort) than 3D visualization.
- Configuration data introduced;
- 2D visualization tool filled with various configurations.

Tool requirements

To be determined in next phase: M12 – M24.

Follow up (M24 / M36)

Give focus on 3d viewer instead of 2d viewer.

3.2.3 Activity A3: Functional Requirements Analyzing and Formalization using DSL

The reason for the new technology

Ambiguities and inconsistencies in today's (M0) written specifications lead to mistakes in the design of products. In the current engineering process these mistakes are found in a late stage (verification and validation). Quality improvement of specifications leads to a reduction in corrective actions in a project and to better predictability of the project's throughput time.

The stakeholders of the new technology

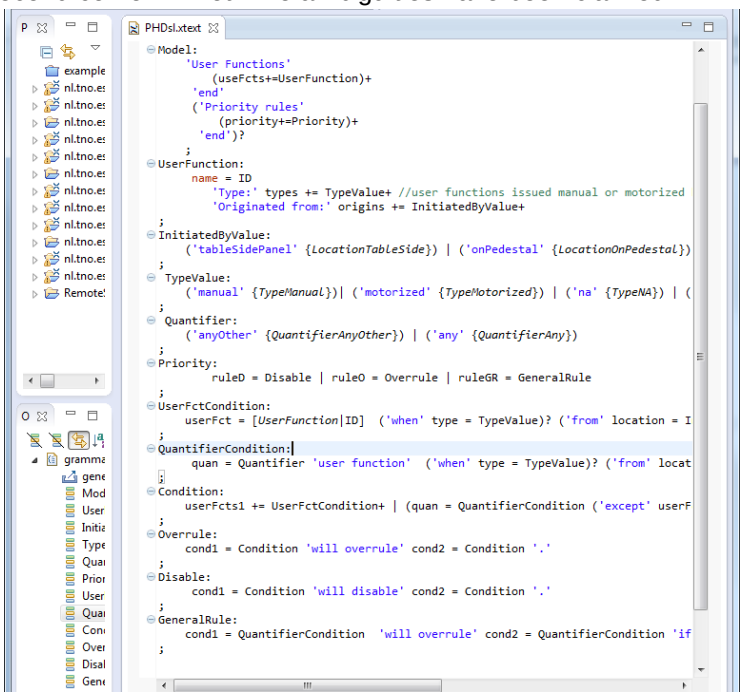
The stakeholders are system designers and software architects.

Approach

The positioning system related function "priority of movements" was selected to evaluate the proposed new method to formalize and analyse requirements specifications. This functionality is described in multiple documents: requirements, design, implementation etc. In order to verify the alignment between these documents we use XText technology to define a DSL for a specific case of priority of movements.

Activities during last year

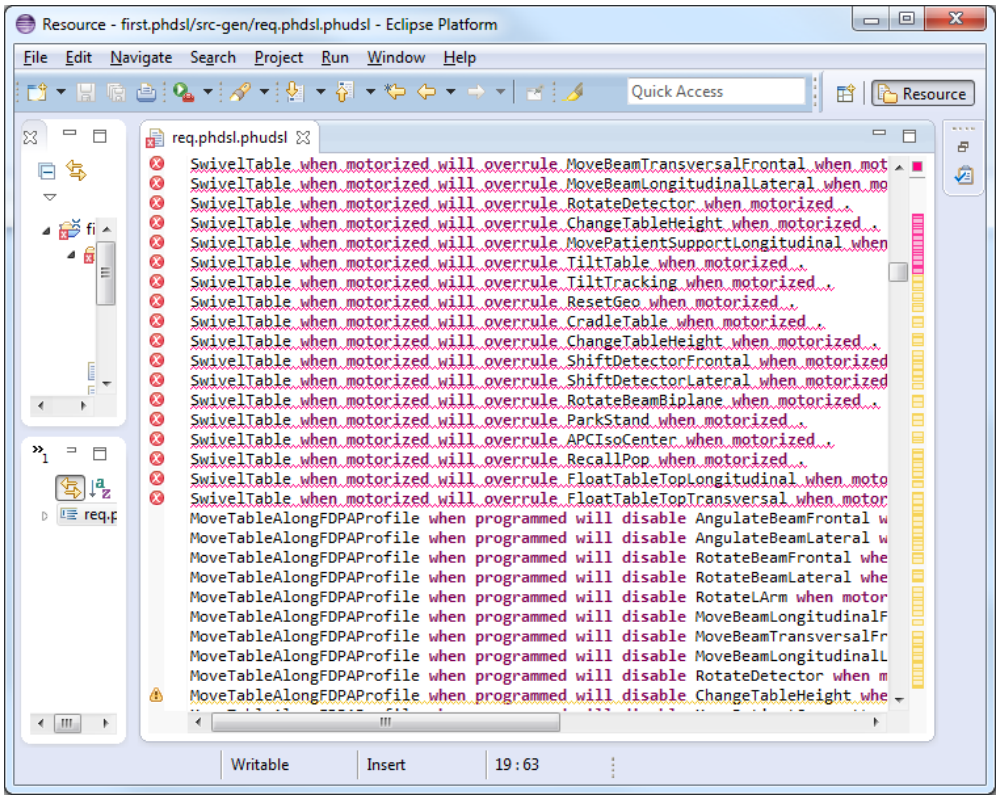
As a first step we used the documents to identify the main concepts. During this phase, we already highlighted ambiguous usage of natural language. For example, most of the movement rules were expressed based on its active state, without defining clearly when a movement is in state active (when user requests the move, when the machine is performing the move, when the movement is scheduled to be performed etc.). Another example, is the use of same term with different meanings: two moves have the same priority means, once, that they are independent of each other, and , in another part of the document, that the first come will be performed and the second come will not. The ambiguities have been clarified through meetings with relevant stakeholders and the concepts have been narrowed to a minimum set.



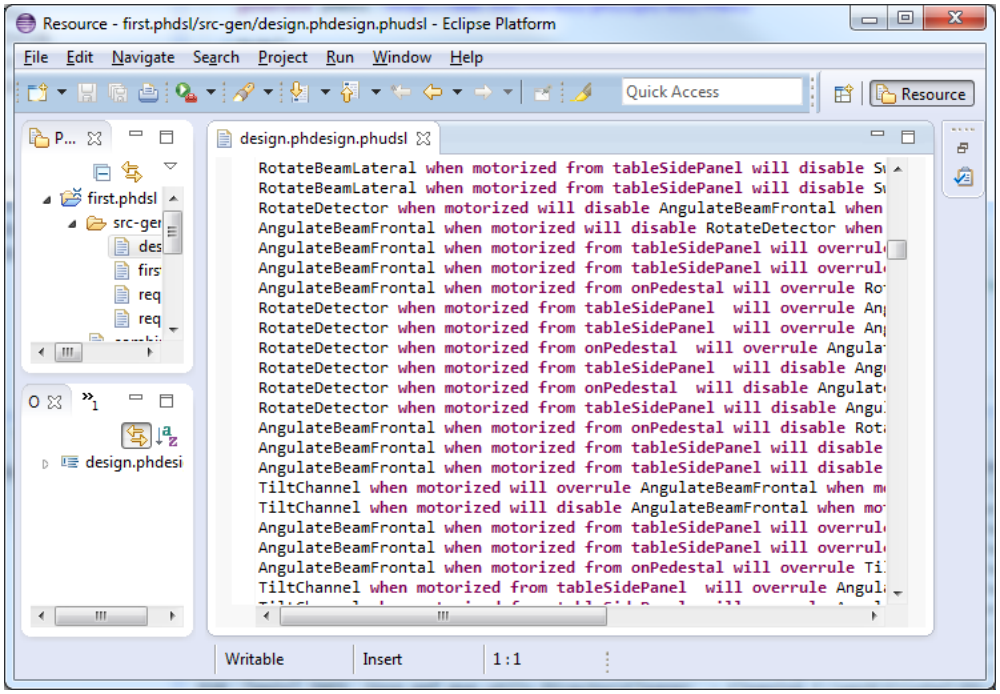
We constructed a DSL using the identified main concepts. Below figure contains a screenshot of the grammar:

The requirements document and the design document have been mapped to the formal language and, in isolation, analysis was performed. The performed analysis is identification of contradicting, duplicate and/or unspecified rules.

The requirement document contained a high amount of rules that didn't pass the analysis part. The figures below shows some of these rules underlined with a red colour.



On the other side, the design DSL passed the analysis without yielding any violation:



Next step was to compare the set of rules from the design and requirements languages and we concluded that the sets have common elements but neither is a subset of each other.

Conclusions

The investigation showed that:

- Translating natural text to a formal language triggers clarification of concepts
- Use of formal language allows checking of correctness criteria in an easy manner and allows to check alignment between different interpretations/documents

Follow up (M36)

- Code generation based on the identified DSL
- Choose which DSL to use for code generation
- Translate also the implementation code into the same language and verify its alignment
- Extend the usage of DSL to other requirements part
- Link DSL to requirement management tools, POOSL and code
- Create an automatic connection between requirements, design and code using DSLs

3.2.4 Activity A4: Infrastructure for early visual verification using a 3D virtual reality viewer

The reason for the new technology

Development of a demonstrator using NobiVR for virtual reality visualization based on a Philips tool (XPoser) implemented using the Ogre3D engine for the 3D visualization.

The stakeholders of the new technology

- XPoser developers
- XPoser users
- PS-Tech

Activities (in Crystal context) during last year

First of all, preparations were made to develop NobiVR from an internal tool to a brick suitable for use by external parties. This involved refactoring/cleaning up the API interfaces of NobiVR, limiting ourselves primarily to the visualization aspects of NobiVR, leaving the tracking input interfaces for a later stage.

Next, NobiVR needed to be extended to integrate with Qt-based applications. As NobiVR was based on the GLUT library, and relied on creating the OpenGL context and window itself, this needed to be refactored and partly re-implemented to integrate with a library such as Qt which creates the window/widget and OpenGL context, in case of the QGLWidget. This development has resulted in the creation of a QtPVRWidget class which is a subclass of the QGLWidget. NobiVR parses its display configuration file and creates the QGLWidget with a QGLFormat based on these settings to enable the desired stereo mode (Figure 11).

```
/* Configuration file of PVR display */
/* Main entries are optional */

mode
{
            PVR_STR_HILACED PVR_DOUBLE PVR_DEPTH PVR_RGBA
            PVR_STENCIL PVR_ALPHA
    /* list display modes:
            PVR_RGBA PVR_INDEX PVR_SINGLE PVR_DOUBLE
            PVR_ACCUM PVR_ALPHA PVR_DEPTH PVR_STENCIL
            PVR_MULTISAMPLE PVR_STR_QUAD PVR_STR_SPLT
            PVR_MONO and/or PVR_REFLECTED
            (REFLECTED if monitor has been adapted
            to display L-R reflected)
    */
}

position
{
    bottom_left -0.2370 -0.1485 -0.5
    bottom_right    0.2370 -0.1485 -0.5
    top_left    -0.2370 0.1485 -0.5
    /* Display corners in Standard Workspace coordinates */
}

window
{
    position    0 0
    size        1280 1024
    fullscreen  0
/* Position and size of window in display */
}

name            XPoser-NobiVR
/* Display to render to */
```

Figure 11: Example NobiVR display configuration file with sections for display mode, display (physical) 3D position, and window position/dimensions.

Once setup, the QtPVRWidget performs the following actions for each frame:

1. Render each viewpoint (1-N)

For example in regular stereo modes such as quad-buffered or interlaced stereo mode, two viewpoints are rendered; the left eye and the right eye.

a. If necessary for the stereo mode, prepare a framebuffer object to render offscreen
b. Setup the viewpoint projection/modelview matrices based on the 3D display coordinates, window coordinates, and head position
c. Render the scene

2. Compose the viewpoints according to the stereo mode (optional)

   For example in interlaced modes, the left and right eye images are rendered from the framebuffer object to the even and uneven lines of the display buffer

Qt applications which use NobiVR need to subclass the QtPVRWidget, and implement the abstract function pvrRenderFunc() in which the scene is to be rendered. Typically, the utilized engine's render function would be called in this function.

As XPoser utilizes the Ogre3D engine, an OgreWidget based on QtPVRWidget has been implemented to integrate Ogre3D/Qt with NobiVR. To integrate a Qt widget with Ogre3D, the engine needs to know that it should use the existing window handle and GL context instead of creating its own. To achieve this, the OgreWidget initializes the Ogre3D RenderWindow as shown in (Figure 12). The existing window handle and OpenGL context handle are passed to Ogre3D during the creation of the RenderWindow, along with the instruction to leave OpenGL control to the application.

```
73      // Get the parameters of the window QT created
74      Ogre::String winHandle;
75      Ogre::String glContextHandle;
76  #ifdef WIN32
77      // Windows code
78      winHandle += Ogre::StringConverter::toString((unsigned long)(winId()));
79      glContextHandle += Ogre::StringConverter::toString((unsigned long)(wglGetCurrentContext()));
80  #elif MACOS
81      // Mac code, tested on Mac OSX 10.6 using Qt 4.7.4 and Ogre 1.7.3
82      Ogre::String winHandle  = Ogre::StringConverter::toString(winId());
83  #else
84      // Unix code
85      QX11Info info = x11Info();
86      winHandle  = Ogre::StringConverter::toString((unsigned long)(info.display()));
87      winHandle += ":";
88      winHandle += Ogre::StringConverter::toString((unsigned int)(info.screen()));
89      winHandle += ":";
90      winHandle += Ogre::StringConverter::toString((unsigned long)(this->parentWidget()->winId()));
91  #endif
92
93
94      Ogre::NameValuePairList params;
95  #ifndef MACOS
96      // Tell Ogre what the window handle is and that the GL context is created and controlled by Qt.
97      params["externalWindowHandle"] = winHandle;
98      params["externalGLControl"] = "true";
99      params["externalGLContext"] = glContextHandle;
100
101     try
102     {
103         qDebug() << "Creating Ogre render window";
104         mWindow = mRoot->createRenderWindow( "OgreWidgetRenderWindow",
105                                              this->width(), this->height(),
106                                              false,
107                                              &params );
108     }
109     catch(Ogre::Exception& e)
110     {
111         qDebug() << "Ogre::Exception: " << e.what();
112         throw;
113     }
```

Figure 12: Ogre3D RenderWindow initialization using existing window and GL context

To be able to render to Ogre3D's render window, the widget needs to define one or more viewports, each with a camera. As NobiVR takes care of the setup of viewports and projection parameters, the widget simply takes the parameters set by NobiVR in the OpenGL state, and feeds these to Ogre3D. Since it is inefficient to do this every frame, the widget maintains a list of viewports and cameras.

Ogre3D cannot render to framebuffer objects created outside of the engine, as it unbinds them before rendering the scene. To utilize Ogre3D created framebuffer objects, the virtual function

QtPVRWidget::createFBO() is overridden in the OgreWidget to return an Ogre3D created framebuffer object. This framebuffer is then used by QtPVRWidget to render the viewpoints to the framebuffer object depending on the stereo mode.

Results

By replacing the OgreWidget used in XPoser with the NobiVR version, the XPoser application now supports configurable VR stereo rendering. Shown in the Figures is the XPoser application running with two different display modes selected in a configuration file without recompilation.

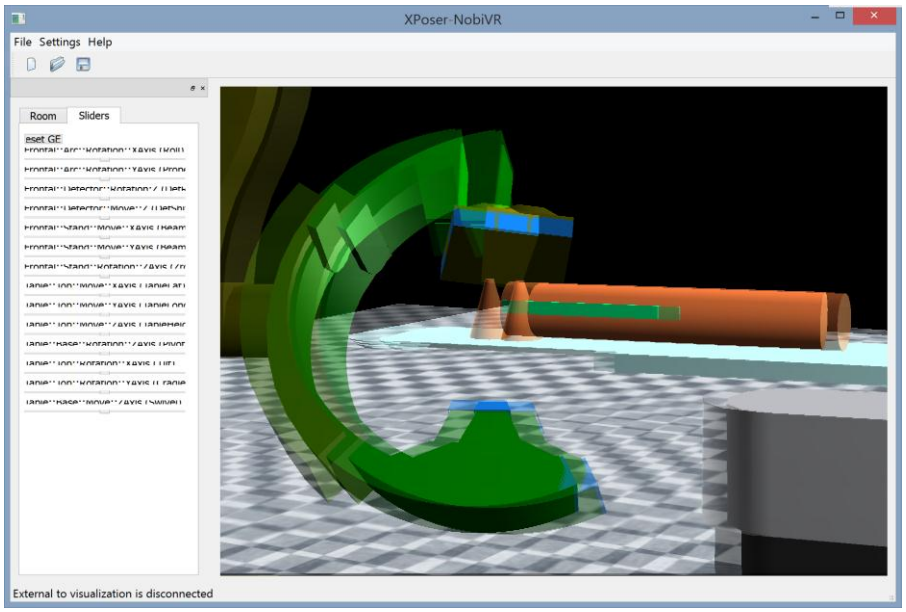


Figure 13: XPoser Ogre3D/Qt application running on NobiVR configured for a desktop interlaced stereo environment (left and right eye images are interlaced line by line horizontally).

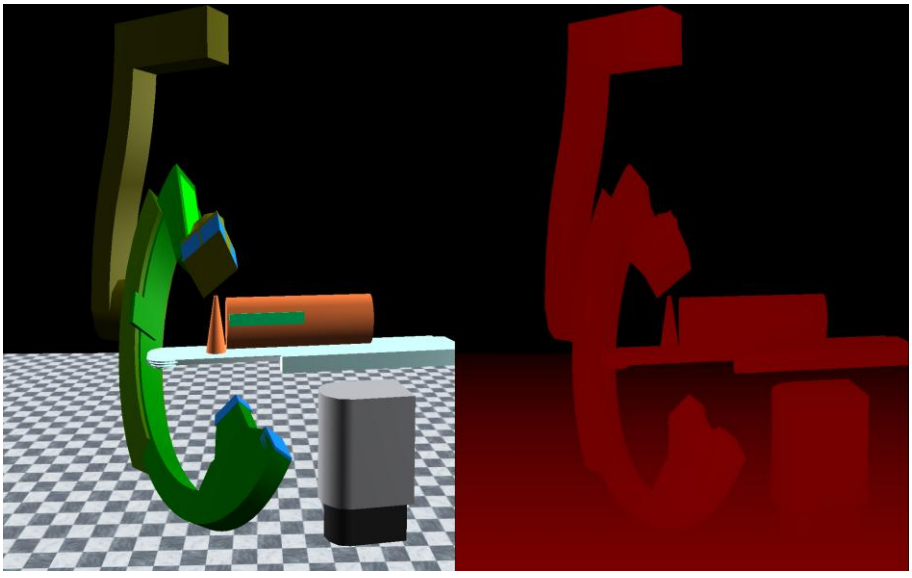


Figure 14: XPoser Ogre3D/Qt application running on NobiVR configured for a Philips WoW autostereoscopic environment (2D image + depth).

Status: what is finished, what is left?

Finished: First milestone in the preparations of NobiVR to develop it into a brick suitable for use by external parties; configurable stereo visualization and physical screen configuration.

Finished: Integration of NobiVR with Ogre3D/QT.
To-do: Virtual reality input devices / head tracking integration.
To-do: Implement display mode(s) for head mounted displays
To-do: Finalize integration with Philips XPoser tool.

Experiences, what was learned?

During the cross-organization and cross-expertise integration process PS-Tech learned more about the specific requirements on NobiVR as a brick for third party use.
Investigation of the Ogre3D engine was necessary to learn more about the supported features of Ogre3D relevant to stereo visualization / virtual reality use, making clear what needed to be added to integrate NobiVR in an Ogre3D application.

Follow up (M24 / M36)

- Increase robustness/stability of NobiVR.
- Further extend NobiVR integration to support virtual reality input devices.
- Increase general applicability of NobiVR layer; extend support to more types of 3D applications.
- Add support for head mounted displays

3.2.5 Activity A5: Early visual verification of formal requirements in DSL using 3D viewer

The reason for the new technology

Visualized requirements offer big advantages over requirements in words:

- Images (especially graphical models) are easy and fast to understand for all disciplines involved in product development, from medical users to architects to designers. They are – in a way - a universal description language for products and their behaviour.
- It is difficult to see through a specification in words; what if scenarios, end of range behaviour and configuration aspects can be understood much easier by all stakeholders.

This is especially true for user needs specifications, used by application specialists (representatives of the medical user), architects and (software) designers. Proper communication about user needs and their consequences for the product will improve the project predictability and the user satisfaction.

The stakeholders of the new technology

The stakeholders are application specialists (representatives of the medical user), architects and (software) designers.

Approach

One important part in interventional X-ray is the eye-hand coordination. The surgeon guides the hand based on the display imaged. Traditionally, an X-ray imaged is always shown: head up, left side of the patient at the right side of the image. This image does not reflect the actual position of the patient on the examination table.

To explore the possibilities of image display an iterative process has been performed using 3D visualization with XPoser and DSL methodology with XText. The stakeholders were clinical application specialists and system designers.

Activities during last year

In the first iteration XPoser has been updated to allow users to visualize the image on the screen based on the positioning system of the machine. We enabled users to manipulate the image orientation. Below a screen shot of this initial exploration is shown:

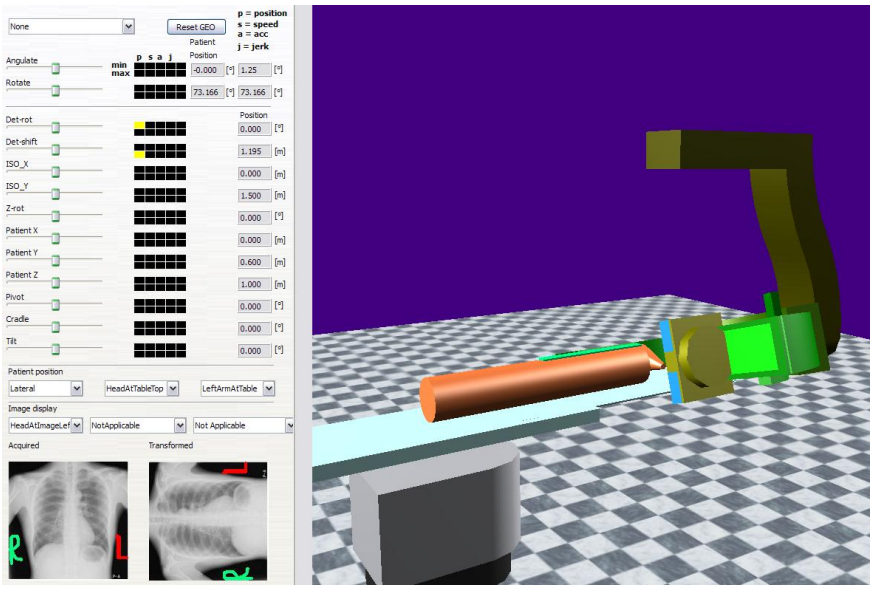


Figure 15: a screen shot of the initial exploration.

Clinical personal was introduced to this approach and asked to review it. The feedback was that two possibilities are the most likely to be preferred (traditional image positioning and orientation of the image based on surgeon position with respect to the patient examination table) and that the user should be exposed to as little as possible parameters. The visualization was updated to incorporate the received feedback: users have the possibility to choose which procedure they are performing and only relevant parameters will be shown. Below the outcome is shown.

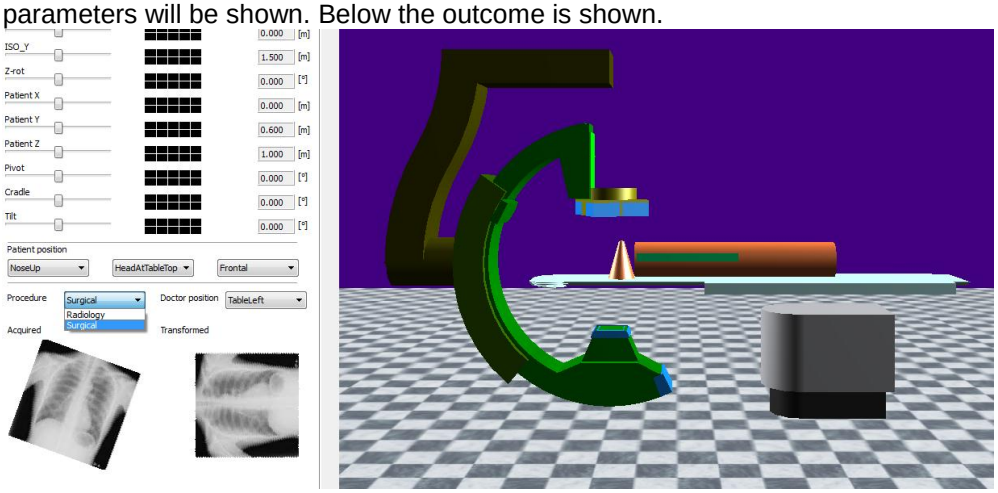
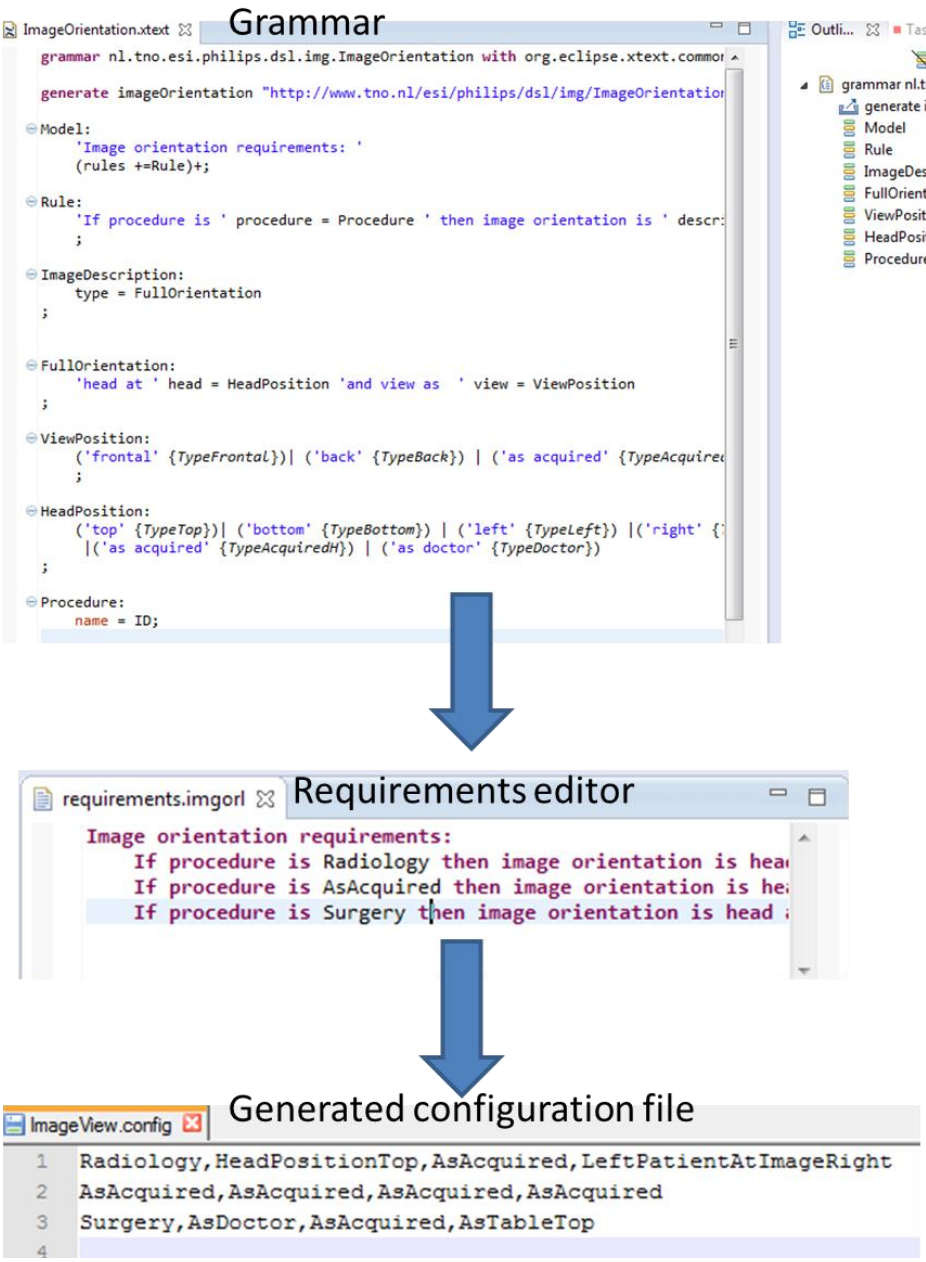


Figure 16: Updated visualization with incorporated feedback.

The third and current iteration coupled the visualization with XText technology to allow easy definition of requirements that automatically can generate new configuration files. These files are then used for the visualization to show the requirements.

XText was used to define: a grammar for specifying the image orientation related requirements, allow users to define new requirements and generate configuration files.



At the same time, the visualization allows for easier customization and display of multiple image views. If a new configuration file is generated at XPoser runtime, the file can be reloaded through the menu *Refresh Image Configurations*.

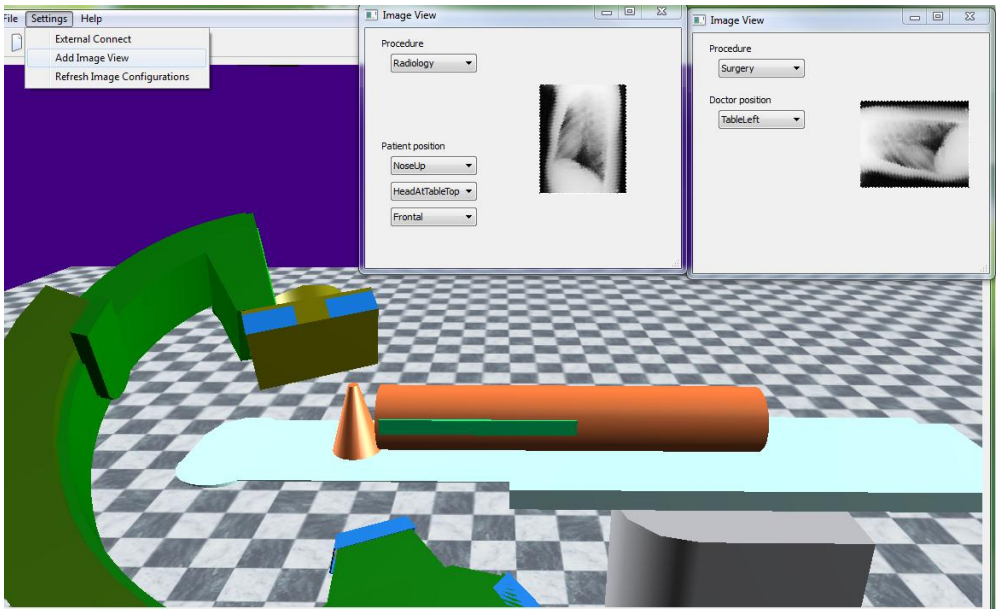


Figure 17: the visualization allows for easier customization and display of multiple image views.

Results

The investigation showed that:

- Visualization allows exploration of user needs in an easy and fast manner
- DSLs methodology is useful only when requirement space is narrowed down by clearly identifying the concepts and their relations
- When DSLs grammar is close to natural language, it becomes easy to use for less technical users
- Automatic generation of configuration files from DSLs allows for straight away visualization of the requirements and easy to reconfigure demo
- DSLs need to allow flexibility for further exploration of requirements and for future re-use

Follow up (M36)

- Present the solution to multiple stakeholders and analyse the acceptance
- Extend the usage of DSL to other requirements part
- Link DSL to requirements management tools and code
- Create an automatic connection between requirements, design and code using DSLs

3.2.6 Activity A6: Couple DSL to requirements management tooling using OSLC

Introduction

TNO-ESI contributes in the Crystal project with, amongst others, a Domain Specific Language (DSL) that allows expressing requirements and their validation and verification. The scope and application of the DSL is the development, by Philips Healthcare, of an interventional X-ray product. Though an extensive set of requirements exists in natural language, important safety aspects of the product have essentially a precise logical nature, e.g. conditions or events that should never occur simultaneously. To overcome ambiguity, redundancy, incompleteness, and many other natural language issues, a formal language with a well-defined syntax and semantics is introduced for specifying the behavior of the X-Ray system (that language is known provisionally as "the DSL").
The problem addressed by the present study is: whereas natural language requirements have a place and a role the systems engineering life-cycle, how can a similar place and role be achieved for the DSL? The working assumptions of the study are that the goal is to establish OSLC integration of the DSL, and secondly to experiment the ideas with the IBM Rational tool-set available to the Crystal project.

The reason for the new technology

The problem addressed by this experiment is:

1. integrating the DSL artefacts and the process of formally analysing specification into the engineering life-cycle with OSLC services such that other roles than requirements analysts can link and trace their work to DSL work
2. scaling up the individual development environment of the DSL workbench to a team-based environment with support for multiple 'threads of work' on DSL's shared among a team of DSL analysts as is usual and desirable in an industrial engineering environment
3. this technology also serves as a prototypical implementation of a first custom made tool chain integration driven by a specific usage scenario from Philips Healthcare, demonstrating the usage of a commercial OSLC solution with bespoke model driven tooling, thus contributing to the Crystal RTP

The stakeholders of the new technology

Stakeholders are:

1. primary: DSL analysts

2. secondary: consumers of DSL work, like designers, testers, architects, managers

Approach

IBM tools considered are

- Rational Team Concert (RTC), a Change and Configuration Management tool;
- Rational Doors Next Generation (DNG) a requirements management tool,
- Rational Design Manager for Rhapsody (DM) a model management tool.

These tools are built on the Jazz(TM) middleware that implements a number of OSLC services.

TNO ESI / eclipse.org tools considered are:

- Eclipse Kepler Standard Package plus XText plugins
- DSL Plugin

Working options (though not entirely mutually exclusive options) for achieving OSLC integration of the DSL are:

1. treat the DSL artifacts as source files (only) and put them under software configuration management with Rational Team Concert (RTC); RTC change sets can be linked to OSLC resources that consume the OSLC CM services

2. treat the DSL as a design domain and create an ontology in Rational Design Manager; the DSL artifacts will then be treated as semantic models, and can be linked to OSLC resources that consume the OSLC AM services
3. link the existing DSL artifacts to natural language requirements in Rational Doors Next Generation by providing an OSLC extension to the DSL eclipse tooling that consumes OSLC RM services

Ad 1 Since the DSL is developed and used entirely in an eclipse (Kepler) environment it is a straightforward operation to bring the DSL eclipse projects under Jazz Source Control. This simplicity comes at the price of functionality: whereas for many programming languages there are compare merge utilities that operate on Jazz Change Sets (the deltas to original file versions that can be treated as an atomic unit), for the DSL there is at this point just a plain text RTC compare/merge utility. Secondly, since links are to change sets, and not to semantic units, the meaning of a link from a DSL change set to specifically a natural language requirements (which is a semantic unit), is somewhat unnatural: typically many, more than one, DSL change sets 'belong' to a formalized version of a natural language requirement. This leads to questions like: how do we know no links have been missed? do we need to recreate all the links of a predecessor on the successor version? (probably, yes). Some of these issues can be glided over by procedures and practices. One could for instance agree to not link change sets, but only components, or baselines of components. The granularity problem re-occurs, however, because there still is a semantic impedance mismatch between RTC components and DNG requirements.

The primary advantage of this approach however is, that it is for the DSL user quite natural for the declarative nature of the DSL to treat the DSL as code. Linking change sets to e.g. Tasks, Defects, Test Cases, Planning and also to design and code artifacts addresses a significant problem: the problem of scaling up working with the DSL in a larger team over a larger number of versions of the product. Questions like who does, or did, what, when and why, and what happened next, are hard to answer with just the DSL alone, but using RTC in the way outlined here makes those questions transparent with OSLC services.

Ad 2. Design Manager stores and operates on models like UML, SysML, BPMN, on a Jazz server. DM is a toolkit that allows for the definition of a modeling domain and its tooling. The abstract syntax of a domain is defined with OWL ontologies. The concrete syntax (notation) and tooling aspects of a domain are defined with DM tooling-specific ontologies. The option 2 above boils down to constructing a new DM domain for the DSL and have custom model constraints and form editors that allow editing of the DSL ontology instances.

The key feature of this approach is that it has the potential to add semantics (constraint checking) to the DSL source that is stored. Therefore the assumption is that this approach could lead to mitigation of the impedance mismatch described above under ad 1, while preserving all of the advantageous features for life-cycle integration of option 1 above.

The downside of this approach could well be that achieving a full OSLC representation of the DSL could imply replacing the client side eclipse DSL tooling from TNO ESI. That is not only a significant amount of work, but might be undesirable to begin with. This needs to be explored in the experiments under this study.

Ad3. Since the DSL tooling is entirely based on eclipse tools (Xtend, Xtext), it is also possible to create in Java an OSLC extension to consume RM services from DNG to link to the DSL artifacts. in the eclipse DSL tooling. This however is only half of the story since it does not bring the DSL in OSLC managed persistence but it provides a native 'requirements look-up' facility in the DSL eclipse tooling. For linking there needs to be a persistent anchor on the DSL side that can hold

the OSLC url's pointing to the DNG side. In order to make the DSL persistent in OSLC there needs to be some OSLC provider mechanism on the DSL eclipse tooling side. This could, hypothetically, be the option 1 or option 2 provided mechanisms (if not excluded by their further design). This needs to be explored in the experiments under this study

Activities during last year

The need for this work has been identified in March 2014 and has been tasked out to IBM with support from TNO ESI to be conducted in the context of their OSLC based Rational tool-set. The working hypotheses of the study have been agreed between members of the WP4.01 team. An experimental infrastructure has been set up in the dedicated Crystal environment at the IBM Amsterdam location. Initial datasets have been created, and work on analysing the scenarios has been started. Scoping of the development of OSLC adapter(s) has been initiated.

Conclusions

At this very early stage no conclusions can be drawn.

Follow up (M24 / M36)

A fully working experimental environment needs to be completed for running realistic scenarios in both using the DSL for product engineering, and for further development of the DSL tooling itself.

3.2.7 Activity A7: Early verification of system design concepts using 3D viewer

For more information about frontal stand and patient examination table see section2.2.

The reason for the new technology

In the process of developing new ideas, at some point an idea has to be made visible in order to discuss the value of the idea with other people within the organization. Within 'Interventional X-ray' projects, there is a good cooperation with Philips Design and often new ideas are visualized by appealing figures such as the one below.

Figure 18: Philips design image (partly blurred)

In the past a next step usually consisted of creating a first functional model of this idea. Such a functional model requires an extensive design effort, development time and budget. When the functional model is seen and tested by colleagues from the marketing and/or application department, more than once the feedback has been …

- "This is not how I understood it, this will not work …"
- "Ah, this is what you meant. I think it will work better if you change …"
- "If I perform this movement, the C-arc is kicking my legs / hitting my head / blocking my view as an operator, … This is not feasible!"

During the past years, an additional step has been introduced. Together with the creation of design images (or after), also animations of the new concept idea are created. The animation is a movie that typically shows a use case performed by the new concept. If useful to explain the full width of the concept, more than one use case can be animated.

The stakeholders of the new technology

The practice of using animations is used successfully within Interventional X-ray during the last years. In general any engineer involved in generating new ideas or concepts can use animations as a means to visualize his ideas for presentation and discussion with other people within the organization or externally.

Activities during last year

The latest animations have served different purposes:

- For clarification of the benefits of a new suspension concept in comparison to the existing product, animations have been created of a specific use case of both the new concept and the existing product. These animations have been used in our own marketing department.

- In order to get valuable feedback from key customers that participate in certain conferences, animations of a new suspension concept have been created and used to explain the concept to the customers. When the customer understands the concept, feedback is asked concerning the advantages or disadvantages he sees in using such a concept in his interventional or surgical labs.
- For a really different suspension concept, some differences in the mechanical design dimensions yield differences in the application use. These differences normally are hard to explain in a short time but the use of animations that show the differences makes it much easier to understand. In that way, together with people from the application department, correct design decisions can be made faster with less chance of misunderstanding!

Animation tool

The tool that is used to create the animations is a program called 'anim8or'. It is shareware and it has been selected for a number of very practical reasons:

- The tool is not difficult to use: a few introduction movies are sufficient to get started.
- The tool is free, so no SW license.

Animation sequence

Within anim8or an environment is created where a lot of components are part of. These components have a certain position-relation with respect to each other or with respect to the 'floor'. The position-relations can be made time dependent which causes components to move with respect to each other. The sequence describes these movements in time. By rendering typically 24 images per second movie time, in the end a movie is created.

Movements can be defined by using an anim8or programming language. Making use of interpolation, a movement sequence can be defined in excel and copy-pasted into the anim8or program. Next screenshot shows an example of a movement sequence.

Sequence overview: PCI-TR procedure with Clea-FD20, okt 2013

															fr/sec
movie : frame 0 … 1460															24
			Frontal stand												
			angrot target		POI target		Larm target		Ibrot target		AD7				
nr	dt	time	Carc	Prop	X_poi	Z_poi	SID	Larm Zrot	Spin	sign	pivot				frame nr
0		0	0.0	0.0	-600.0	0.0	119.5	0.0	90.0	-1.0	0.0				0
1	1	1	0.0	0.0	-600.0	0.0	119.5	0.0	90.0	-1.0	0.0				24
2	4	5	0.0	0.0	-120.0	0.0	119.5	0.0	90.0	-1.0	0.0				120
3	0.5	5.5	0.0	0.0	-120.0	0.0	119.5	0.0	90.0	-1.0	0.0				132
4	3	8.5	0.0	0.0	-120.0	0.0	119.5	90.0	90.0	-1.0	0.0				204
5	2	10.5	0.0	0.0	-70.0	0.0	119.5	90.0	90.0	-1.0	0.0				252
6	2	12.5	0.0	0.0	-70.0	0.0	119.5	90.0	90.0	-1.0	0.0				300
7	2	14.5	0.0	0.0	-120.0	0.0	119.5	90.0	90.0	-1.0	0.0				348
8	0.2	14.7	0.0	0.0	-120.0	0.0	119.5	90.0	90.0	-1.0	0.0				352.8
9	3	17.7	0.0	0.0	-120.0	0.0	119.5	55.0	90.0	-1.0	0.0				424.8
10	0.2	17.9	0.0	0.0	-120.0	0.0	119.5	55.0	90.0	-1.0	0.0				429.6
11	2	19.9	0.0	0.0	-120.0	0.0	119.5	55.0	90.0	-1.0	15.0				477.6
12	2	21.9	0.0	0.0	-120.0	0.0	119.5	55.0	90.0	-1.0	15.0				525.6
13	2	23.9	0.0	0.0	-120.0	0.0	119.5	55.0	90.0	-1.0	0.0				573.6
14	1	24.9	0.0	0.0	-120.0	0.0	119.5	55.0	90.0	-1.0	0.0				597.6
15	2	26.9	0.0	0.0	-120.0	0.0	119.5	0.0	90.0	-1.0	0.0				645.6
16	1	27.9	0.0	0.0	-120.0	0.0	119.5	0.0	90.0	-1.0	0.0				669.6
17	2	29.9	25.0	10.0	-120.0	0.0	119.5	0.0	90.0	-1.0	0.0				717.6
18	1	30.9	25.0	10.0	-120.0	0.0	119.5	0.0	90.0	-1.0	0.0				741.6

Animation movie
The output of anim8or is a movie typically showing the new concept in its intended environment, with or without operators. Next figure shows an example.

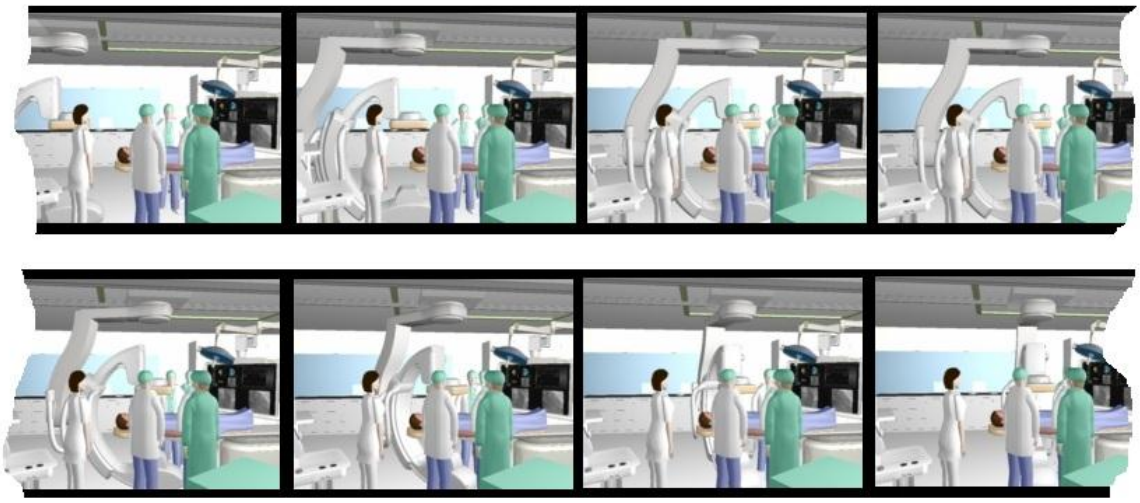


Figure 19: animation movie.

The movie, often an avi-file or wmv-file, can be shown using a standard media player or can be embedded in a powerpoint presentation.

Results

During the last years, animations have been used successfully within Interventional X-ray. Since computing power grows fast, future animation software will contain more and more functionality. If an alternative is needed at some time, the program 'blender' is used by a lot of university students.

After working with animations for several years now, we can state that

- Creating an animation takes quite some time: the environment has to be set-up, the movement sequence has to be defined and implemented, other components have to be moved in or out if needed, a good camera viewpoint has to be determined and good lighting settings have to be found (avoiding annoying reflections on the components). Also rendering the movie(s) can be quite time consuming.
- We learn a lot by making an animation with a new concept because we virtually place the new concept in an examination room together with other equipment and we make it move. By doing so, we experience what is easy and what is difficult and we have to solve the problems that are caused by the new concept (e.g. while programming its movements).
- An animation really can show the benefits or disadvantages of a certain concept in a very small time because it's almost like we actually have a prototype and made a movie with it in a clinical environment performing relevant use cases and showing the interaction with operators or other equipment in the examination room.

The use of animations has some clear benefits:

- More than an appealing figure of a new concept, an animation can 'walk around' a new concept showing it from different viewpoints (not only the most positive one) and an animation can show the movement possibilities of a concept for one or more application use cases, also in a clinical environment.

- Although the creation of animations takes some time, it is significantly cheaper than building a functional model. If the implementation of concepts that turn out to be ‘not feasible’ or ‘I thought it would be different’ can be avoided, a lot of time, money and frustration can be saved.
- It’s easier to take an animation to a customer or an overseas colleague than a functional model.
- Early feedback facilitates design improvements or concept change decisions in the right direction. This significantly increases the chance that the first prototype fulfills the expectations and makes people enthusiastic.

Follow up (M24 / M36)

Most animation programs have more than enough capabilities for creating movies based on moving components. Functionality is only expected to increase. If in the future another animation program has to be selected, blender is considered a possible candidate.

In future video tools shall be integrated in the engineering environment. Reviewing simulated system behavior and modeled scenarios shall be integrated.

3.2.8 Activity A8: Early verification of system design concepts using demonstrator

For more information about frontal stand and patient examination table see section 2.2. More detailed information on Use Case 4.1 is available in [ref1] and in High level description of use case and context.

The reason for the new technology

To validate the high-level patient oriented movement concept, a software demonstrator has been built to demonstrate the usability aspects of the next-generation movement behavior on a current generation X-ray system.

The stakeholders of the new technology

The primarily concerned parties for this visualization are medical application specialists, system designers and software architects/designers.

Approach

For verification of user needs and applicable software concepts with marketing and application a demonstrator has been built. This was realized by adjusting the current software to include a prototype kinematics layer to enables patient-oriented movements.

Activities during last year

- Definition of forward and inverse kinematics formulas
- Implement in legacy software architecture
- Verification / Validation of usability concepts with application experts

Results

Definition of forward and inverse kinematics formulas

The principle followed is to first express the end position of a movement request in the object coordinate system and then determine each x, y, z coordinate separately.

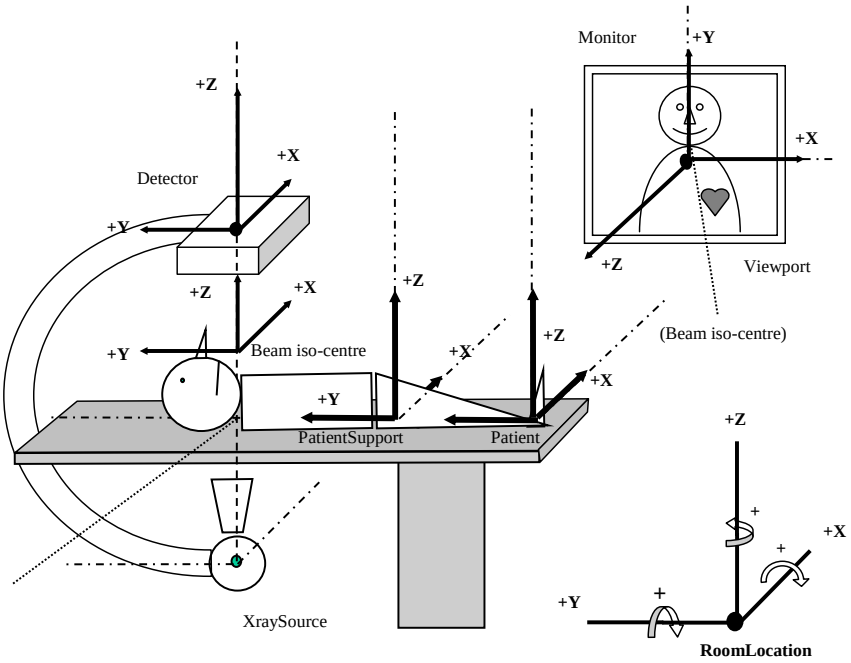
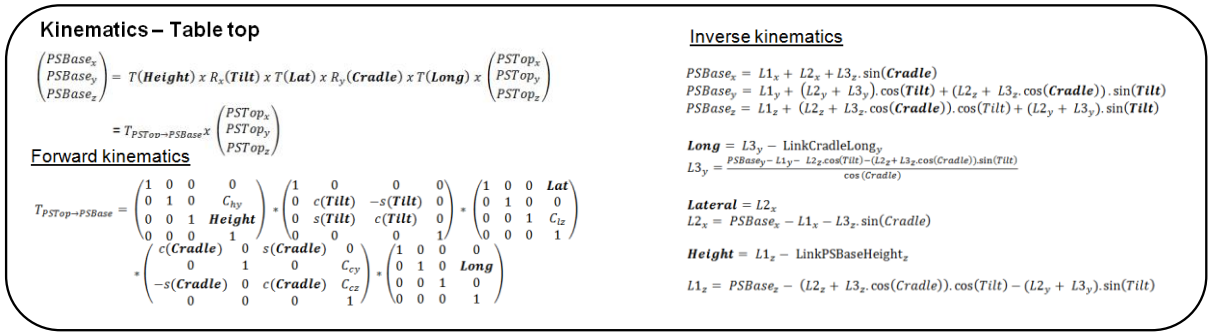


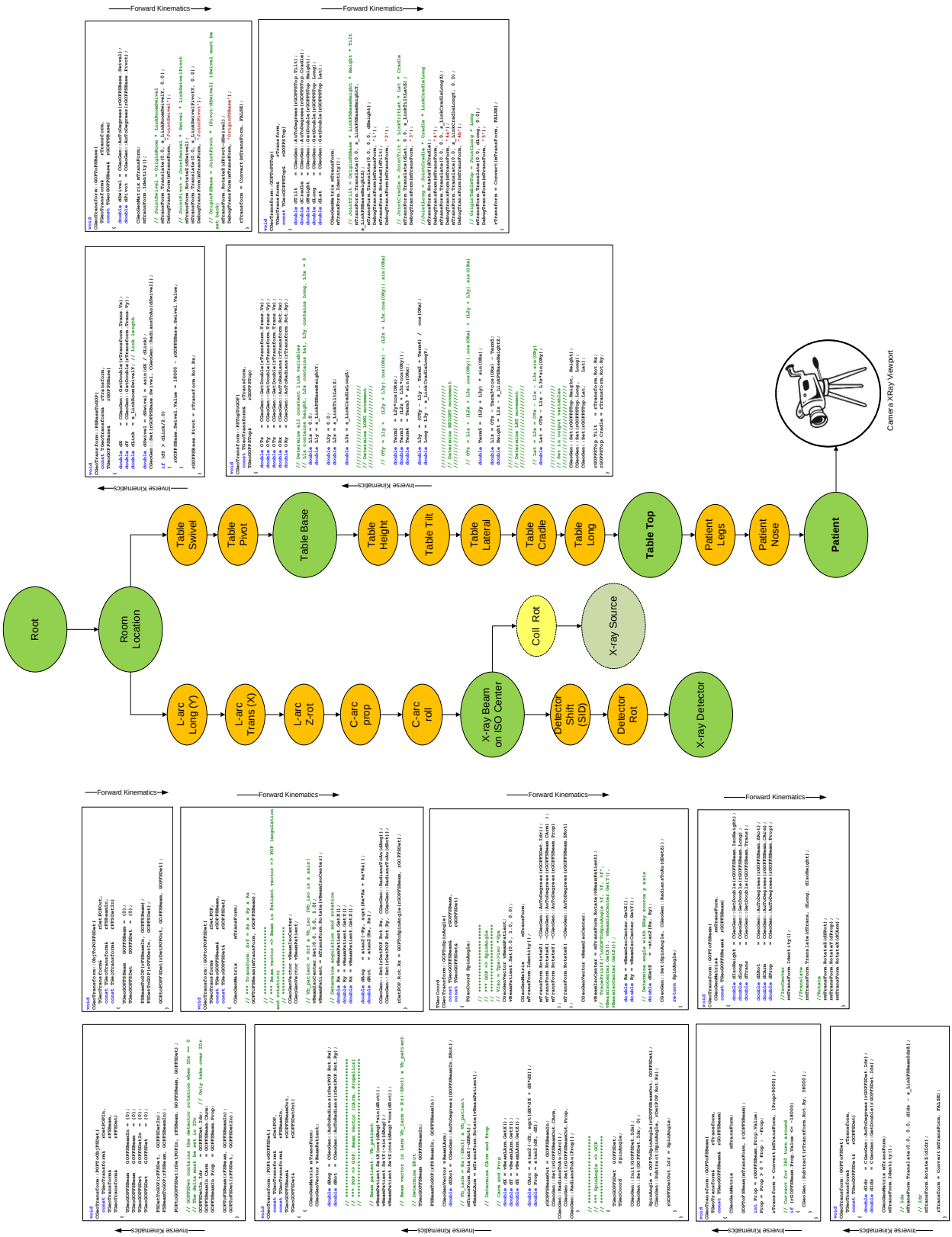
Figure 20: Object coordinate system of the positioning system.

The positions of the GeoObjects[3] are expressed by the position of the GeoObject origin. The position is expressed in the GeoObject parent's coordinate system. In the figure below, the analytical forward and inverse kinematics formulas are shown:



The principle followed is to first express the end position of a movement request in the object coordinate system and then determine each x, y, z coordinate separately. In the figure below, the kinematics model of the C-arm is shown as a scene graph, including code fragments to convert between the different coordinate systems.

[3] GeoObjects are parts of the positioning system, like detector, X-ray source, patient support table.



Verification and validation of usability concept with application experts

Workshops have been performed to verify and validate the usability concept with application experts from the different clinical segments. For review means, we made use of video editing tools to save the workshop outcome.

Follow up (M24 / M36)

The demonstrators are considered to be very useful, but creating them takes too much time. Towards the future we should use models to speedup process of creating demonstrators.

3.2.9 Activity A9: Early verification of mechatronics design concepts using Matlab

The reason for the new technology

In next generation Interventional X-ray machines there is an increasing need to provide the physician with simple means to control complex movements of the system. For example, a physician is typically only interested in the angulation/rotation values[4] of the image that is shown on the screen. When moving the system using a joystick, the physician wants to manipulate the angulation/rotation values of the system directly.

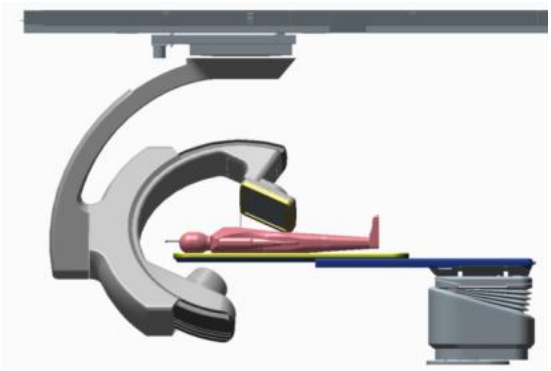


Figure 21: Case 1: The rotation is manipulated via the propeller axis. The angulation is directly manipulated via the roll axis.

The requirements of the movement, such as speed, acceleration, braking path, etc., are typically specified on the level on which they are experienced by the physician. As a consequence these requirements do not apply directly to the mechatronic system that is being created by the designer. Moreover, due to the fact that multiple axes are moving at the same time, they experience reaction forces from each other, which leads to very complex calculations.

To overcome these difficulties a Matlab model was developed that transforms the so-called patient-oriented movements that the physician experiences to the axis-oriented movements that the engineer has to work with. Moreover, the model also calculates the reaction forces that the different moving axes exert on each other.

The stakeholders of the new technology

- System architects; define the patient-oriented requirements
- Mechanical/mechatronic designers; use the outcome of the model for their design

Approach

The model consists of two parts:

1. Translation from patient-oriented movements to axis-oriented movements
2. Computation of required forces and torques

The first step of the model is created using kinematic relationships. The user enters the desired parameters on patient level into the graphical user interface, see figure below.

[4]Angulation values go from head to toe while rotation values go from left to right.

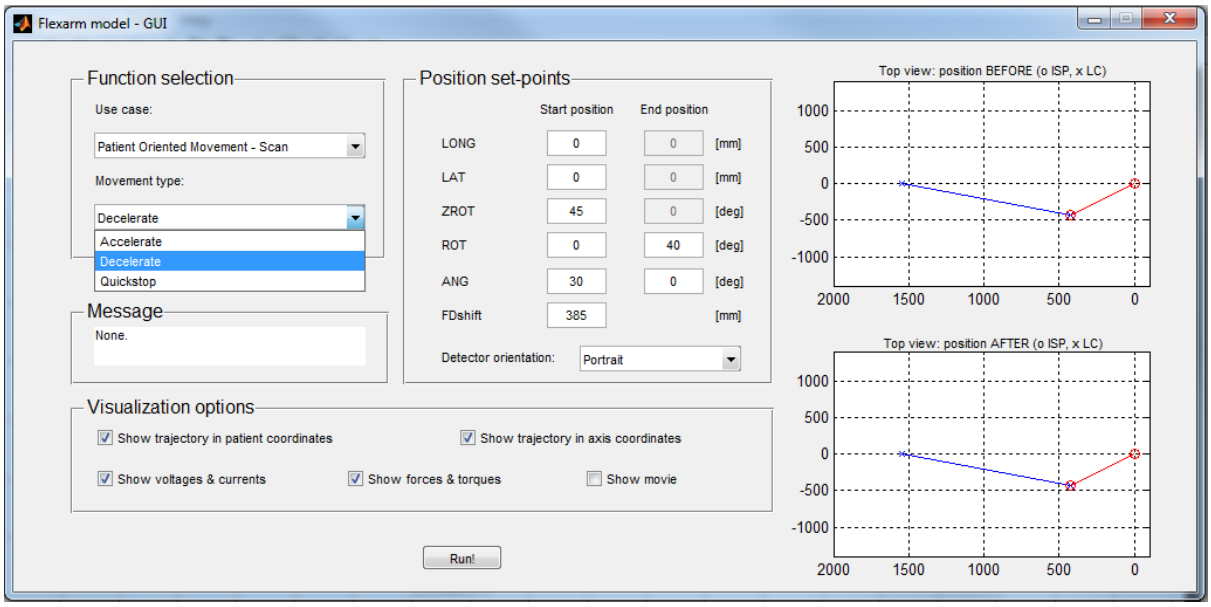
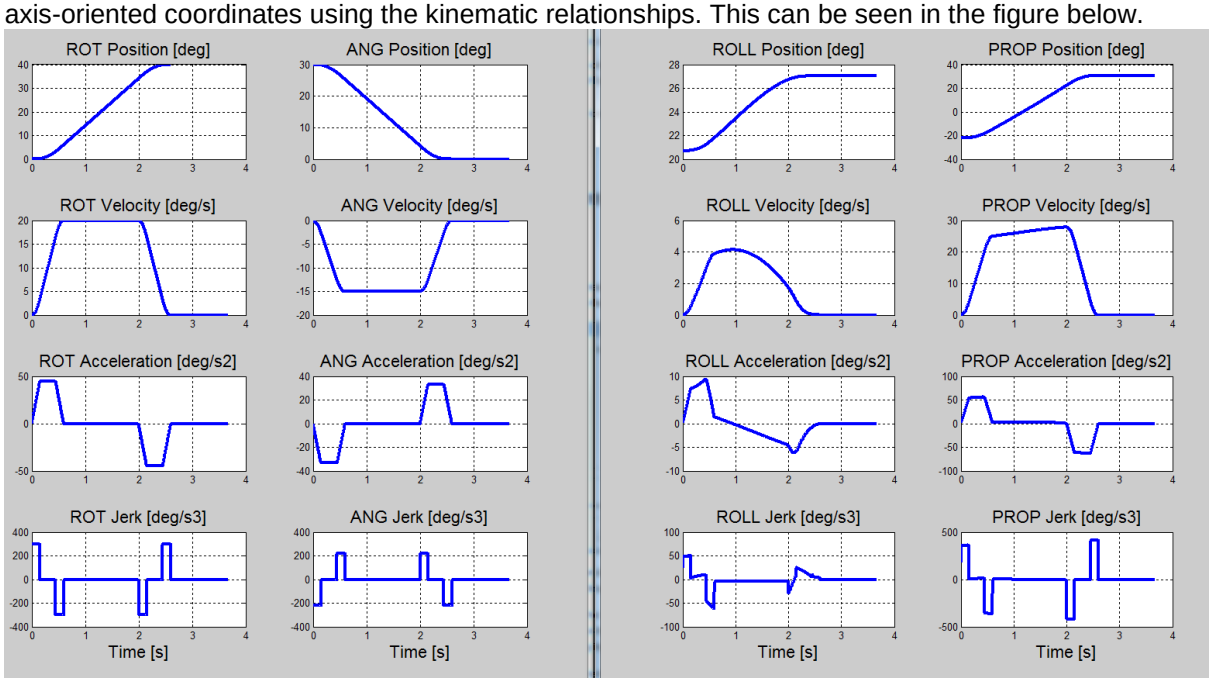


Figure 22: The graphical user interface of the model, which allows choosing use-case parameters and the desired output.

These parameters are used to create trajectories in the patient-oriented coordinates and then translated to axis-oriented coordinates using the kinematic relationships. This can be seen in the figure below.



The desired trajectory in patient-oriented coordinates (i.e., rotation and angulation values) which is based on the requirements (e.g., movement speed and acceleration) from the system architect.

The translation to axis-oriented coordinates, which provides requirements that can be used by the mechanical/mechatronic designer for his or her design.

Note that the translation is non-trivial, e.g., the ROT and ANG speed of 20 degrees per second gives rise to a PROP speed of almost 30 degrees per second. This kind of information is very hard to obtain without a structural modeling approach that involves kinematic relationships.

```
27 -    if max(abs((LAT+0.6.*sin(-ZROT))/1.2))>1
28 -        error('Impossible combination of LAT and ZROT provided');
29 -    end
30
31 -    RZ1 = asin((LAT+0.6.*sin(-ZROT))/1.2);
32 -    RZ2 = ZROT - RZ1;
33 -    LC = LONG + 1.2.*cos(RZ1) + 0.6.*cos(RZ1+RZ2);
34
35 -    ret_code=[LC;RZ1;RZ2];
```

Figure 23: simplified example of the equations involved in the kinematic relationships.

The second step is to compute the forces and torques that are needed to generate the axis-oriented movements and hence also the patient-oriented movements. These computations take the reaction forces of the several axes into account. The computations are based on rigid-body equations.
The rigid body equations can be used to describe the output forces and torques (in six degrees-of-freedom) of a rigid body based on the input forces and torques and the external forces and torques that act on the body. This together with the equations is illustrated in the figure below.

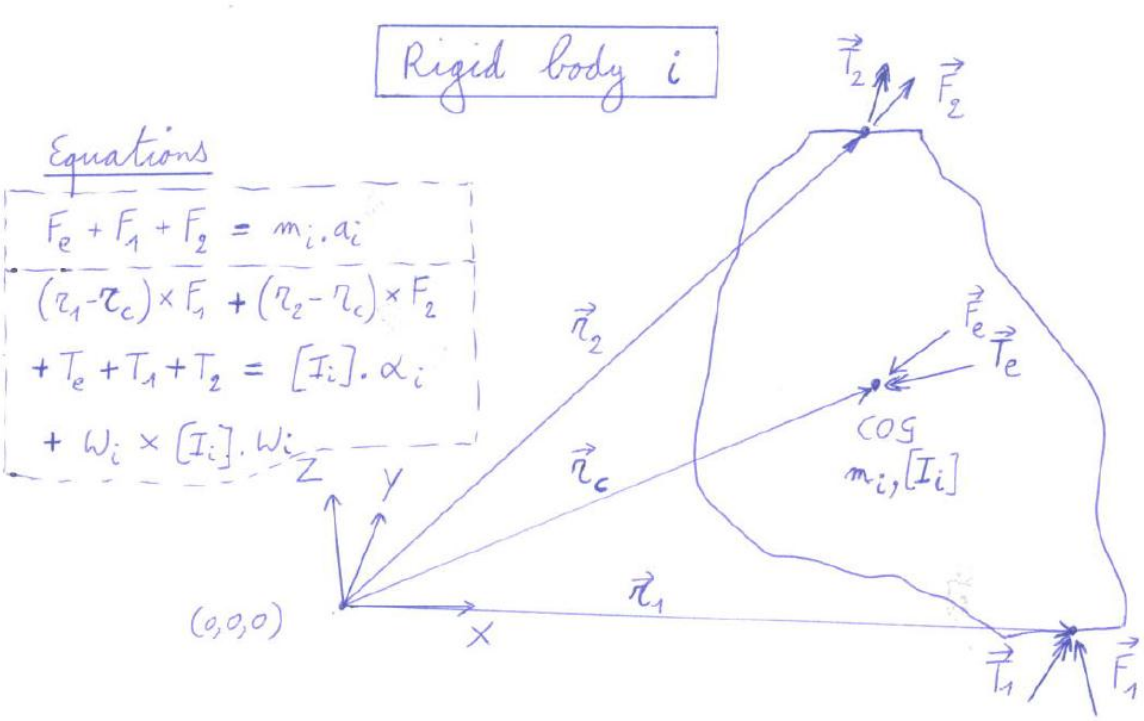


Figure 24: Rigid body equations for a single body. Indicated are the input, output and external forces and torques together with the distances to a reference point.

When several of these rigid bodies are connected to each other properly and the right mass, inertia matrix and center of gravity are specified for each of these bodies, it is possible to compute the forces and torques in the system.

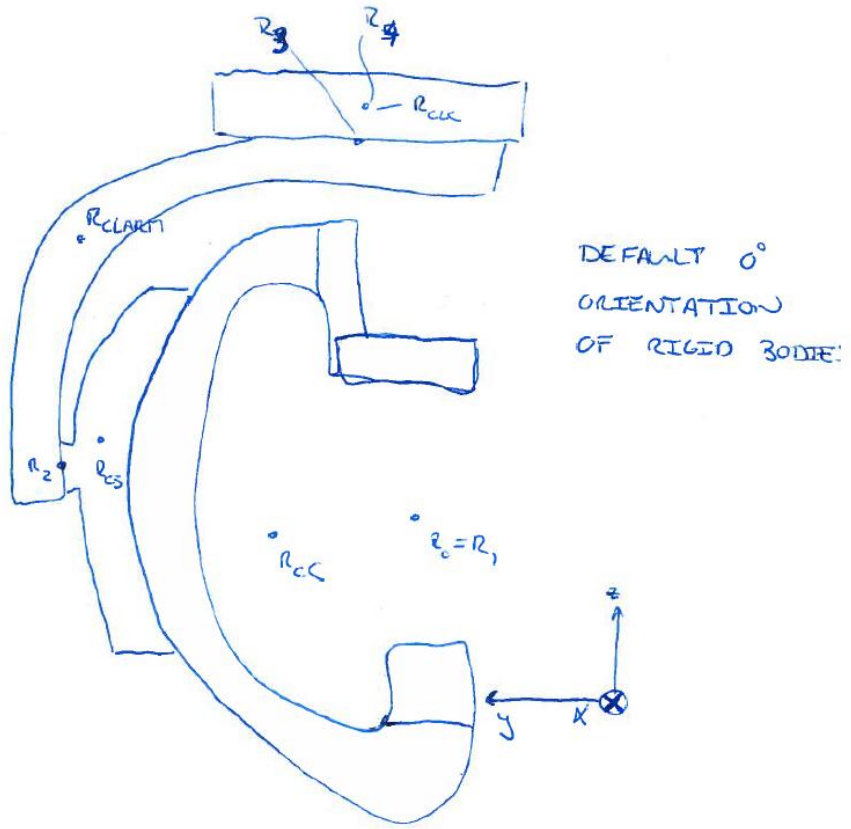


Figure 25: A connection of five rigid bodies that together make up for a complete interventional x-ray system. Indicated are the locations of the center of gravity and the interface points where the rigid bodies are connected to each other.

The result of such a computation is shown below.

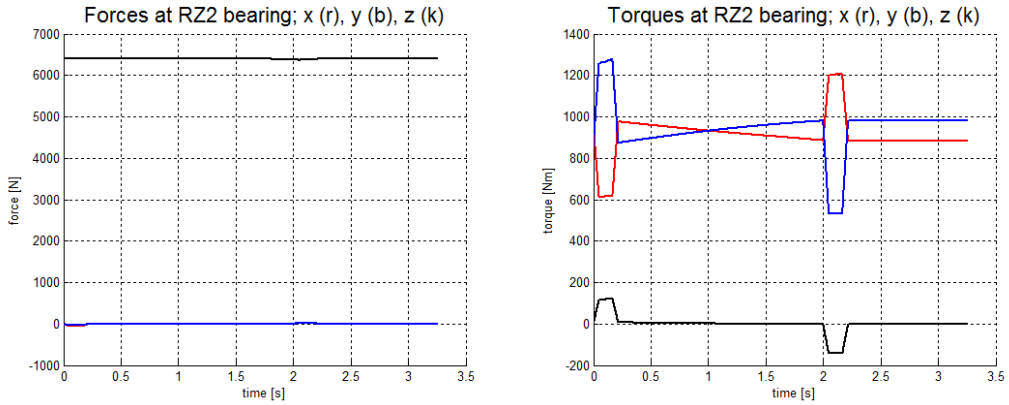


Figure 26: result of a complete run through the model. Shown are the forces and torques in six degrees-of-freedom at a certain interface point.

These forces and torques can then be used by the mechanical/mechatronic designer to choose the power of the drivetrains and the strength of the mechanical components.

Activities during last year

During the last year the entire tool has been implemented for two cases. More specifically the following actions were performed:

1. Requirement definition together with system architects

2. Derivation of the kinematic relationships
3. Derivation of the rigid body equations
4. Development of the graphical user interface
5. Implementation of the kinematic relationships
6. Implementation of the rigid body equations

Results

The model was developed and implemented for two different types of patient-oriented movements in the past year. The outcomes of the model have been used in the design and verification of the machine in one of the two cases. The mechanics for the other type of patient-oriented movements are currently being designed. Furthermore, discussions have been started to make the model suitable for a third type of patient-oriented movements.
Keep in mind that each new type of patient oriented movements requires a new set of requirements and kinematic relationships. However, the rigid body equations can be re-used almost completely. When a new positioning system is studied it is possible to re-use the structure of the rigid body equations, one merely has to specify the changed center of gravity, etc..

Follow up

In the next iterations the model will be refined and extended to include effects that were previously not included. Moreover, the model will be adapted to other system geometries and other types of patient-oriented movements. Moreover, the accuracy of the model's predictions will be investigated and improved where possible. As the current modeling approach lacks the advantages of 3D visualization and modeling, we will investigate the possibility to integrate the mechanics 2D model in the 3D visualization.

3.2.10 Activity A10: Early verification of mechatronics design concepts using Matlab and 3D viewer

For more information about frontal stand and patient examination table see section2.2.

The reason for the new technology

The main workflow of an Interventional X-ray system is the same as for any other diagnostic system: the camera and patient shall be positioned with respect to the each other, 2D or 3D Image shall be taken, post processing shall be done, enabling diagnosis of the patient.

An interventional X-ray system is not only used for examinations but also for treatments, e.g. with 'minimal invasive' procedures. As the technology gets more advanced the system is used in a wide spectrum of applications (cardio, vascular, neurology, electrophysiology and surgery) leading to expanding the set of requirements to the system.

Following requirements are leading in the development of the mechatronics and movement behavior of the system:

- Easy, intuitive control
- Flexibility in positioning and set-up of the system
- Safety for patient, operator and equipment

Until now the flexibility of positioning the system was limited and within this limitation axis control was sufficient. The introduction of rotatable beam unleashed the flexibility of the system and enables controlling the beam instead of the axis. Patient oriented movements are introduced and independent of how the system is set-up it can be controlled the same way.

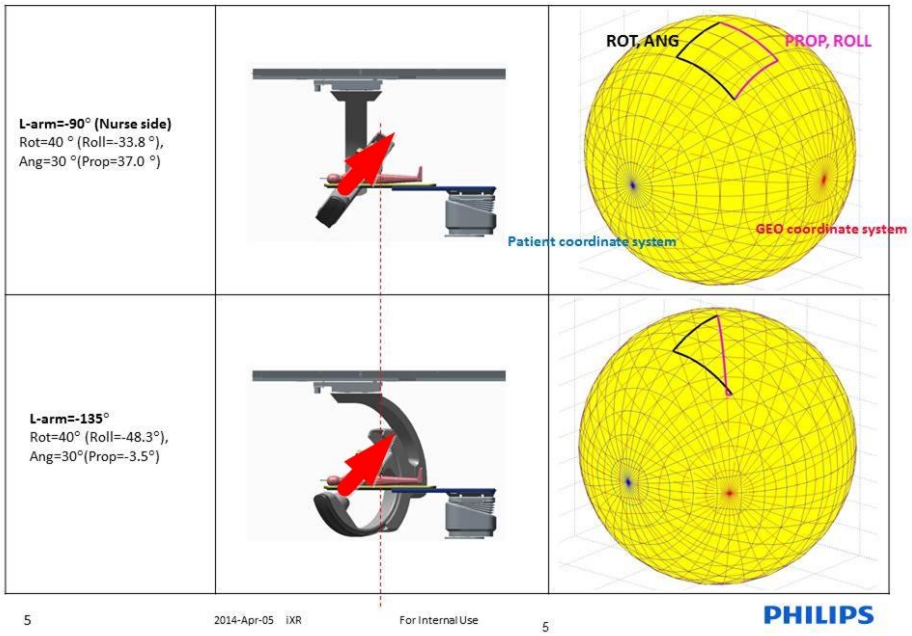


Figure 27: different set-up is shown of the system with same rotation and angulation angles with respect to the patient, leading to different propeller and roll axis movements

Usability is important but not at the expense of safety. With an interventional X-ray system this is a challenge because physician and other operators are close to the patient most of the time. Moreover, an optimum needs to be reached between dose and image quality. Therefore it is important to position the detector as close as possible to the patient, avoiding collisions with the patient.

Now the challenge is to optimize usability, safety and capabilities of such a system?
First of all an overview was needed to handle the number of configurations (stand and tables), set-ups and movements. To create the overview a MATLAB model was developed.
For interaction with the user a hybrid model was developed of MATLAB and Xposer visualization tool.

The stakeholders of the new technology

System architects, software architects, mechatronic architects and clinical users

Approach

Key success factor was going up and down in abstraction level, get an overview but also understand in depth how things work and then move up again to assure simple behavior across the application field of the system.

MATLAB model was created that enabled sweeping throughout the application field. E.g. this enables to find maximum axis speeds, accelerations/decelerations needed for patient oriented joystick movements. But also to see how these patient oriented movements are restricted by mechatronic limits and safety limits.

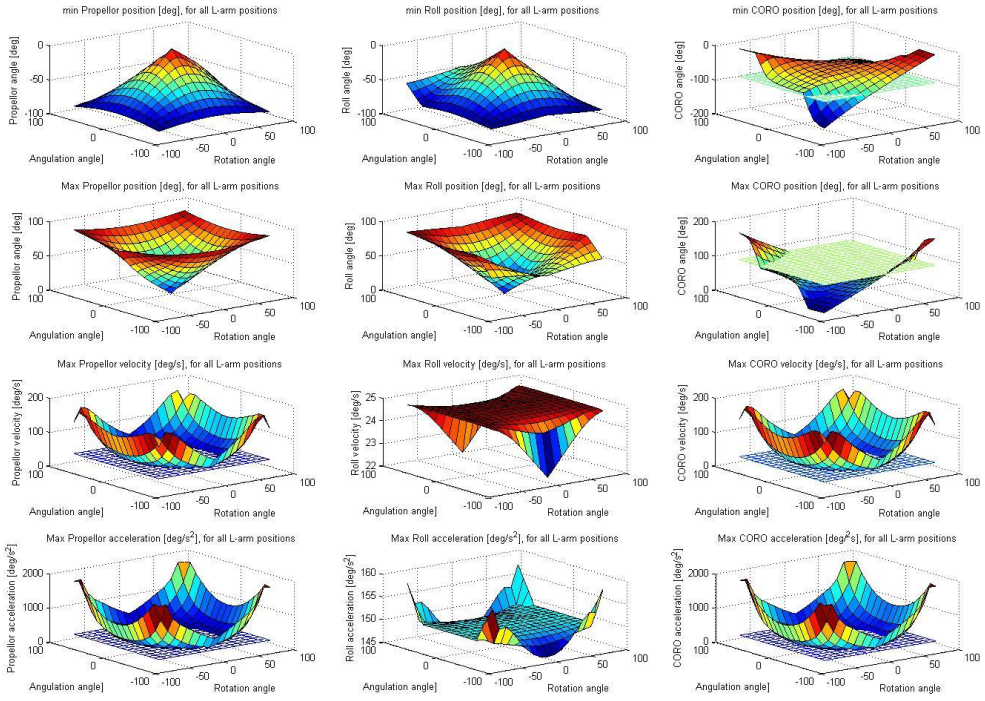


Figure 28: Maximum axis speeds, accelerations/decelerations reached for patient oriented joystick movements

For interaction with the user a hybrid model was developed of MATLAB and Xposer visualization tool.

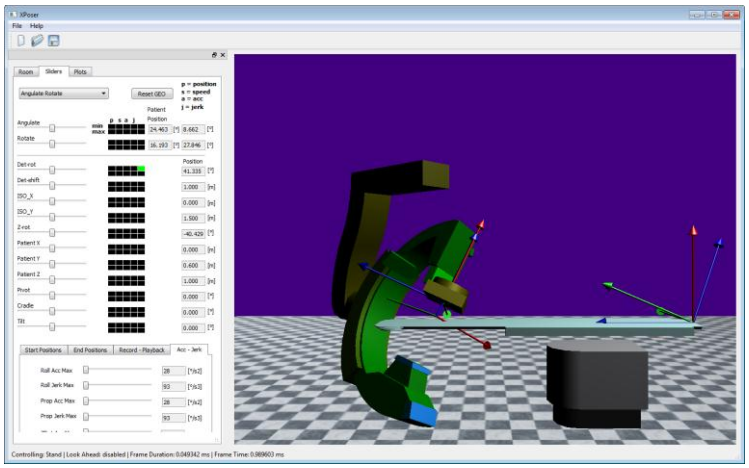


Figure 29: figure shows XPOSER visualization

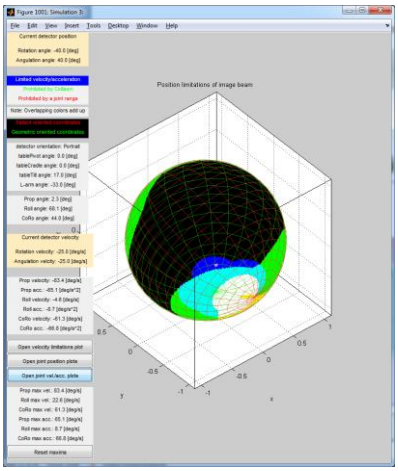


Figure 30: figure shows MATLAB GUI

Visualization
Simulation consists of 2 parts.
1) XPOSER Visualization
- which enables control of movements
- shows actual position of stand and table

2) Matlab visualization
- All joint range limitations, velocity limitations, acceleration limitations (roll, propeller, detector, collimator)
- Collisions (table and stand)
- From several windows can be opened to display detailed information

Setting system positions
The position of the system is defined by the table positions (long, lat, height, tilt, cradle and pivot) and the stand positions (X/Y- Position, L-arm angle and FD shift).

Real time behavior
The patient oriented position of the detector in the simulation can be controlled by a joystick or sliders. The visualization of the sphere and graph has to animate real-time with the machine orientation in the 3D simulation.

Activities during last year

During the last year the all tools described were developed for limited set of

- Configurations → main ceiling stands and tables
- Movements → mainly rotation and angulation
- Safety restrictions → mainly stand –table collisions

Results

The developed models are used for design of a new machine. Models enabled simulation of movement behavior and interaction with clinical users in an early stage of the development. Calculation of restrictions of the mechatronic system on system behavior enabled balancing between usability and cost. Moreover now it was possible to look over entire application field and to define better and more uniform behavior and safety measures.

Tool requirements

We need tool that combines requirement below

- real time visualization of the system and interaction with the user
- enabling analyses of interaction between system behavior and component
- accessible for mechatronic developers

Follow up (M24 / M36)

In the next iterations the model will be refined and extended to deal with a larger set of configurations and concerning more complex movements. As the current modeling approach incorporates 3D visualization and modeling together with a full automated version with the SW end product, it brings much more value than activity A9 does.

3.2.11 Activity A11: Early verification of mechatronics design concepts using demonstrator

For more information about frontal stand and patient examination table see section 2.2.

The reason for the new technology

Major design changes – especially when they are discovered in a late phase of the project - may lead to significant delays and cost increases in a project. In this innovation study, early prototyping is used to decrease the chance of major design changes in the actual development project. More specific 2 early prototypes have been built:

- a one-to-one scale model of rapid prototyping materials
- a motorized prototype

The stakeholders of the new technology

The different forms of early prototyping can be used by innovation, pre-development and/or development teams. When a prototype is available it can be shown to other people within an organization for evaluation.

Approach

One-to-one scale model

The goal of the one-to-one scale model is to evaluate the look and feel of the geometrical design within the environment of the examination room. The degrees of freedom that are important for the evaluation are present in the model, others are left out. For example within the study the degrees of freedom of the suspension were important to evaluate because that was the new part of the complete positioning system. The degrees of freedom of the C-arc itself were not present in the model because their behaviour was already known and understood.

Although in a final design the degrees of freedom or movements are motorized, they were not motorized in the scale model. The movements had to be performed manually. This is sufficient in this stage of the project because the goal is mainly to evaluate the look and feel in different poses of the system rather than the movements of the suspension itself.

Such a model allows evaluating multiple aspects of the design which are important for the colleagues from the marketing and application departments:

- design aspects: bulky?, stable look, appealing, in line with the rest of the system, effect of color, etc. These are a lot of properties that are very difficult to express in numbers and also subject to personal preferences.
- ease of use aspects: are there conflicts with other equipment in the room, does the system provide enough working space for the operators, can the system be positioned where desired, how does the system move through the room (simulated manually), is there enough space for all the operators in different clinical applications.
- safety aspects: which are potential hazards related to this design, how about collisions with other devices in the examination room, can the system be moved out of the way in case of an emergency with the patient?

Motorized prototype

The goal of the motorized prototype in this specific project was to explore the feasibility of the mechatronic control and behavior and the usability aspects of the system. This prototype requires considerably more effort to realize than a one-to-one scale model because it requires a complete mechatronic design including drive trains, guidings, mechanic components with numerous interfaces, amplifiers and controller. The look of this prototype is very technical and not at all according to the design with nicely shaped covers, also no covers were part of the prototype.

Such a model allows evaluating several mechatronic and usability aspects:

- Is it possible to tune a controller for every movement that has enough bandwidth and stability margin? Do the movements interact with each other's motion control: e.g. does the presence of one movement cause instability in the control loop of another movement?

- Is the reproducibility of the movement end positions according to specifications?
- Does the suspension allow for proper calibration related to 3D reconstructions and 3D roadmapping features of the X-ray system?
- Is the control of the system intuitive for the operator?
- Can the system be moved manually? Are the manual movement forces too excessive or not?
- Which buttons need to be provided for local control? Which specific functionality is required for each of the buttons?

Activities during last year

Last year a lot of effort has been spent on the design, assembly and testing of the motorized prototype. Many design calculations have been performed in order to select the proper motors, drive trains and guidings and in order to create mechanical drawings for the manufacturing of parts. After assembly of all parts, connecting to the electronics and setting to work, the process of tuning the individual motion controllers and performing simultaneous movements took place. In a next step a lot of mechatronic measurements for evaluation of the prototype have been performed. Simultaneously sessions with people from our marketing and application departments were held to evaluate the usability aspects of the system.

Results with examples

One-to-one scale model

The pictures give an idea of the materials used in the scale model and of the setting in a test room.

Figure 31: rapid prototyping materials

The scale model has been shown to an extensive group of colleagues but also to various Philips sales representatives from all over the world and to some key customers (doctors). Some examples of very specific and unexpected results of this scale model:

- The sales force has become very enthusiastic about this new suspension and are really pushing the development organization to speed up the development of this suspension.
- The feedback of a key customer based on his experiences with this scale model in an examination room setup, led to a major design change in mechanical dimensions which we were able to implement during the realization of the motorized prototype.

Motorized prototype

The picture shows a part of the motorized prototype and gives an impression about the level of detail.

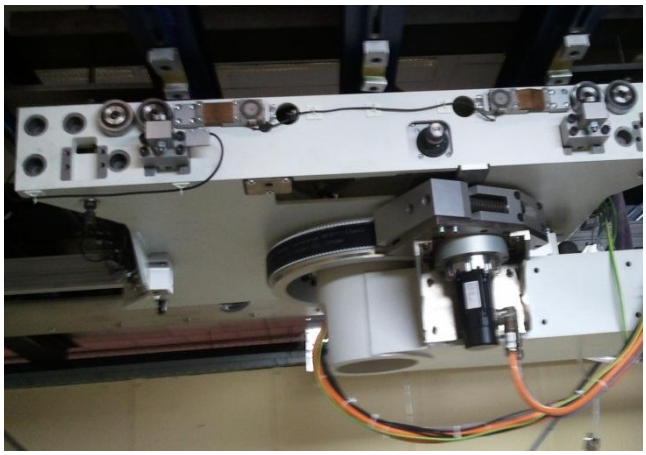


Figure 32: part of the motorized prototype

The prototype is controlled by means of an XPC-target that is connected with the motor amplifiers and position sensors. The setpoint generation and motion control feedback loops are implemented in Matlab-Simulink. After downloading to the real-time XPC-target the movements of the prototype can be controlled through matlab commands. This constitutes a very flexible environment for performing experiments and tests.

Also this prototype has been demonstrated to an extensive group of colleagues, sales representatives, service engineers and some key customers.
The main results of this prototype:

- Demonstration of the feasibility of the motion control, including simultaneous movements.
- Specific test results and learnings concerning the mechatronic aspects of the design. These learnings will certainly be taken into account during the next step design in a development project.
- Demonstration of the ease of use in the controlling of this system. Agreement on the number of button and their functionality to be implemented in the development project.

Status
The main outcomes of these early prototypes are available. In that respect most of the work is finished. The motorized prototype however is still being used for specific tests by the development team.

Experiences
Between the first implementation of a new idea and the final product, some evolution takes place in the form of minor and major redesigns. If this evolution process is allowed by building one or more early prototypes, the likelihood of another major redesign during the development project is decreased significantly.
Next to that early prototypes can make people enthusiastic and allow engineers to perform tests in order to improve the next step design.

Tool requirements

No specific tool is built or needed for early prototypes. It is a certain way of working that can be embedded in the development procedures of an organization.

Follow up (M24 / M36)

See how we can reuse the mechatronic models created for early demonstrator in the rest of the development process. Also investigate the possibility to make demonstrators (using models) based on (parts of) our own system.

3.2.12 Activity A12: Early verification of software design concepts using POOSL

The reason for the new technology

In the medical domain there is a tension between the requested speed of innovation and the time needed to deliver a certifiable system. To ensure the required safety, usually a long test and integration phase is needed. To shorten this phase and to avoid late bug fixing, the aim is to detect faults (if any) much earlier in the development process. In this use case, the emphasis is on early fault detection during the architecting phase.

Traditionally (reference) architectures are described using various informal drawings that are easy to make and modify. However, once the developers start to reach an agreement on such informal drawings, it is still very difficult to decide whether the architecture will really work in practice. Often the only way to decide this is to start implementing it.

In this use case, we consider an architecture for the control of motions of an interventional X-ray system of Philips Healthcare, see Figure 33. It consists of many moving parts such as C-arms, with X-ray generator and detector, and patient examination table. Besides accurate movements for a large range of medical procedures, the architecture has to ensure safety, including collision prevention.

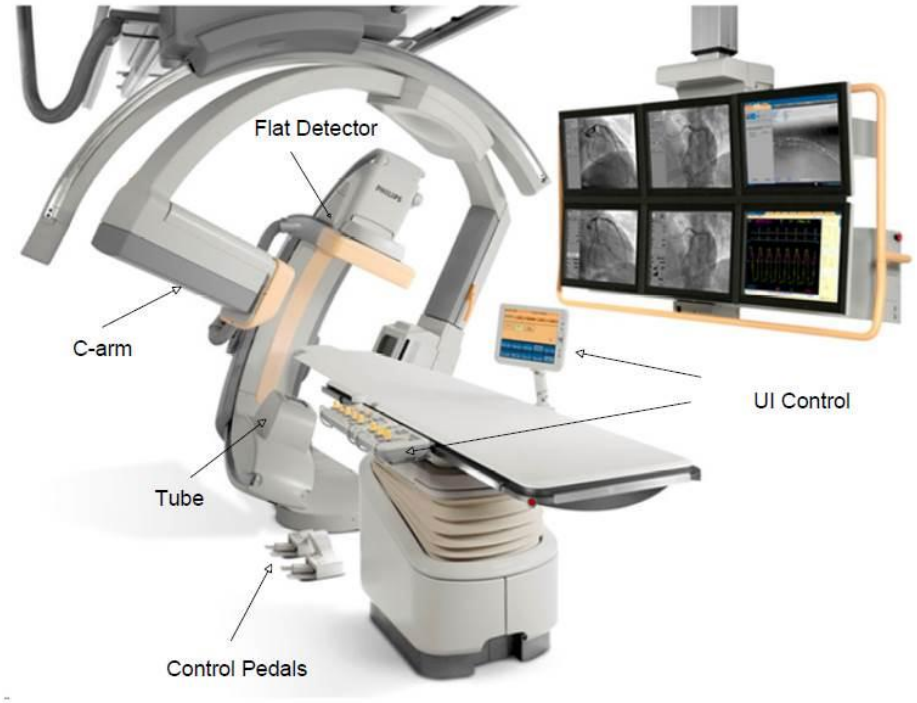


Figure 33: Interventional X-ray system

To migrate from the current legacy code a hybrid architecture is being developed. Following the traditional approach, white board drawings are converted into Visio and PowerPoint pictures, see Figure 34.

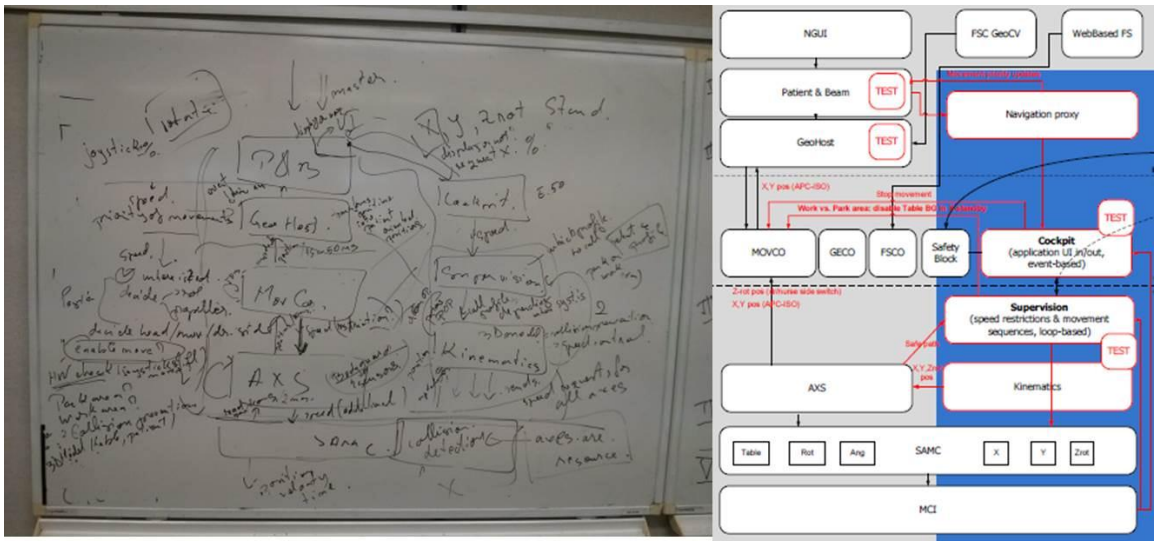


Figure 34: Informal architecture descriptions

Before starting the implementation, we would like to validate the architecture in order to reduce risks. Especially for this complex hybrid architecture it is important to clarify the responsibilities of the parts, the interfaces of the layers, and the data flow. The aim is to gain confidence that several typical scenarios can really be implemented effectively, and to investigate how the architectural choices impact the system as the user would experience it.

The stakeholders of the new technology

Stakeholders for the new verification method are system architects, software architects

Approach

To obtain more confidence in the feasibility of the architecture, we have used high-level formal models, which can be analyzed using simulation and domain visualization (see also references [3], [4], [5]). The approach consist of the following ingredients:

- A visualization of the system input which can be used to insert events and data during model simulation. Typically, this input is stored in a buffer to decouple real-time user input and the execution of the model using simulation time.
- An executable model of the architecture, including the behavior of the main components. To keep the workload manageable, it is crucial to limit the amount of detail and to focus on critical scenarios that have a high impact on the system.
- A visualization of system output to show the correctness of the architecture. Preferably, this visualization shows the resulting user-perceived behavior of the system to allow validation of the main concepts with various domain experts.

To validate the modeled behavior, we use interactive simulation. In our experience, it is important to relate the architectural model to the user-perceived system behavior. That is, not only consider internal software aspects, but also include their impact on the full system. Hence the emphasis should be on visualizations of user interactions and externally visible system behavior.

Activities during last year

The approach mentioned above has been applied to the hybrid architecture of movement control. We have addressed the three ingredients as follows. The three components are connected by sockets: system input, model of architecture and system output.

System input

To stimulate a simulation of the architectural model, we have used several interfaces to simulate user requests for movements. Figure 35 shows on the left a basic QT interface to trigger a number of basic movements. The right part of this figure shows the UI module that is used by the medical staff; a picture of this module has been included in a small Java program which allows clicking on buttons and areas of joysticks, so simulate user actions. We have also experimented with a game controller to request movements.

Figure 35: User input

Model of the architecture

To model the architecture, we used a language called POOSL (Parallel Object-Oriented Specification Language [1], [2], which has a formal semantics defined in terms of a timed probabilistic labelled transition systems. POOSL uses two types of building blocks: cluster and process. Clusters can contain again clusters and processes, and thus they are very suitable to model hierarchical system structures. Processes focus on individual behaviors and are specified using a textual object-oriented process algebra. Each block has an external interface consisting of ports that can be used for synchronous one-to-one message communication; that is, a message can be communicated when a sender and a receiver are both ready for communication.

During the first modeling activities, we have used the SHESim tool which includes an interactive simulator for POOSL, see Figure 36. It can show interaction diagrams with the flow of messages between parts of the model. Moreover, during simulation the internal state of the model can be inspected.

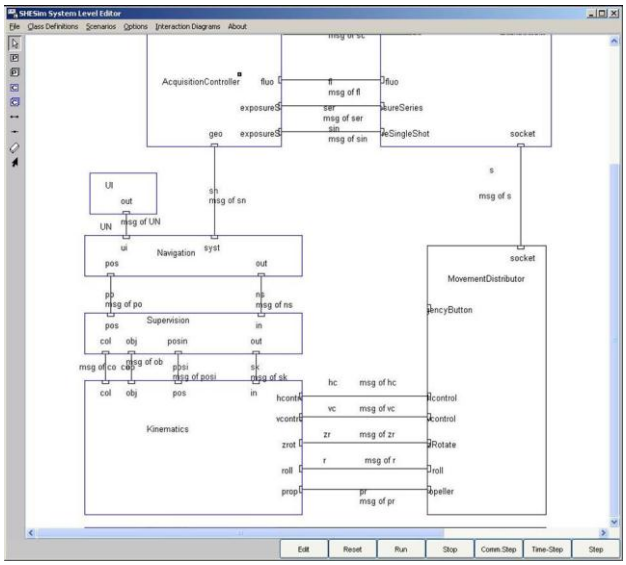


Figure 36: SHESim tool for POOSL

Later in the project, we switched to the new Eclipse development environment for POOSL, as depicted in Figure 37, which is being developed in WP 6.3 of Crystal. This user interface uses the XText/Xtend technology for Domain Specific Languages, as addressed in WP 6.10. It interfaces with the Rotalumis simulation engine.

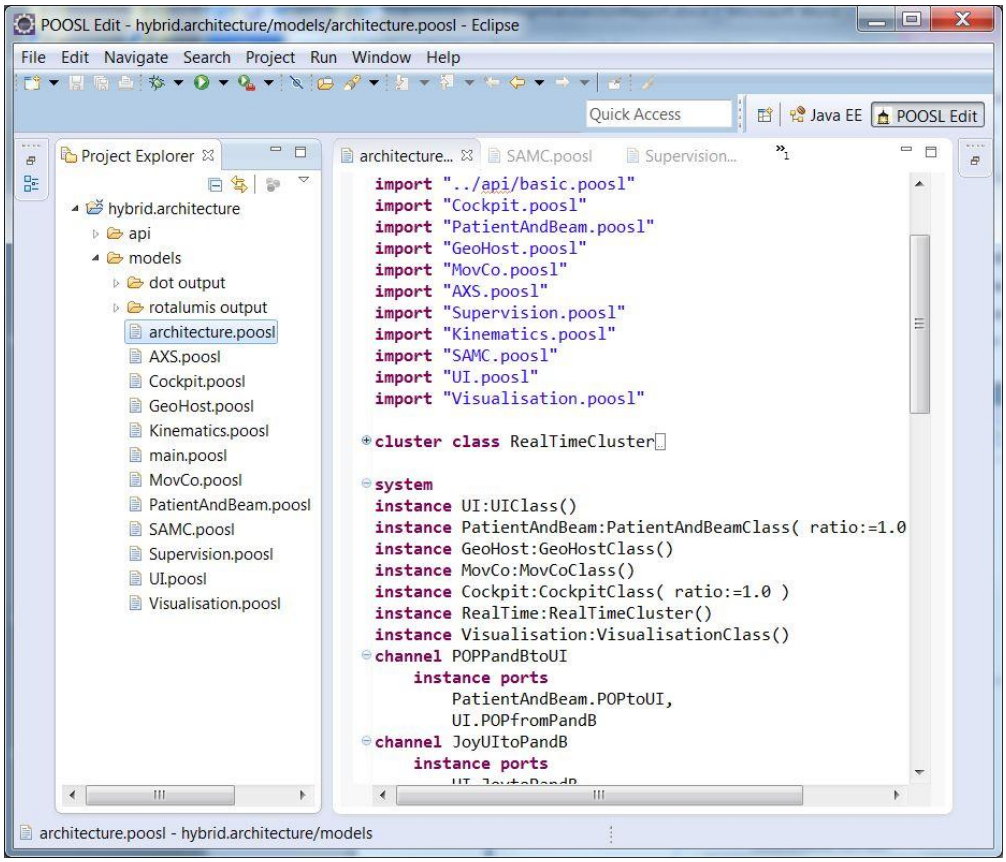


Figure 37: Eclipse interface for POOSL

Note that SHESim models can be imported in the Eclipse user interface and the Eclipse environment allows an export to the SHESim format.

System output
An important aspect of this use case is the early validation of the movements and the ability to assess safety aspects without having to use a real physical system which is expensive and often not available. To this end, we first developed a prototype visualization using Blender [6], as shown on the left in Figure 38. Blender is an open source tool that combines a 3D modeling environment with an interactive game engine. Although this visualization was already very useful to get insight in the details of the control architecture, it was also rather limited. For instance, table movements are very much restricted. Hence we switched to the Xposer model, shown on the right of Figure 38. It is more realistic, allows more movements, and enables different viewing angles. This visualization so based on the open source graphics rendering engine Ogre [7].

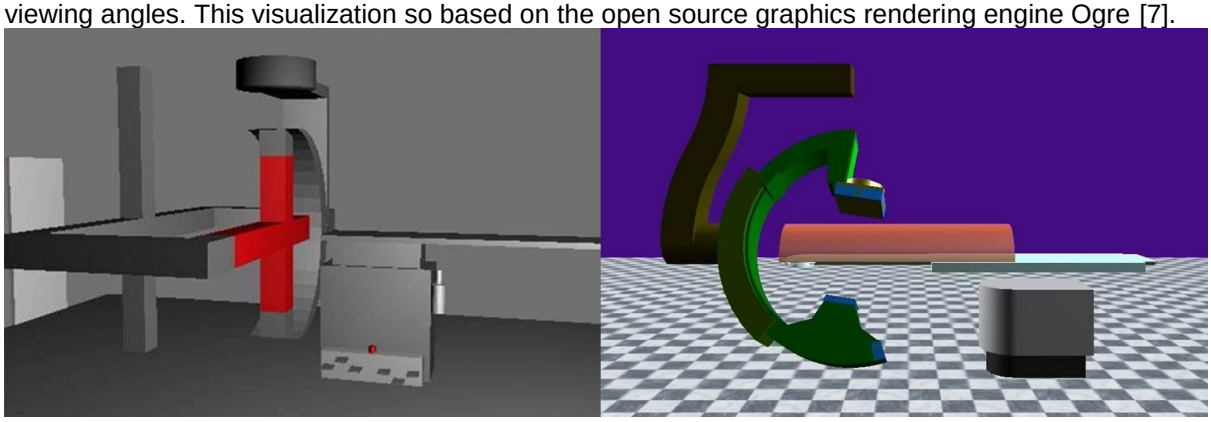


Figure 38: Visualizations of movements.

Results

The activities mentioned above resulted in two instances of our approach to validate software architectures, applied to movement control:

- A combination of a Java user interface, an architecture model expressed in POOSL using the SHESIM tooling, and a visualization in Blender. This instance models a preliminary version of the software architecture.
- A combination of a Java user interface, a POOSL model in Eclipse, the Rotalumis simulation engine, and the Xposer visualization. This version models the most recent version of the hybrid control architecture.

Making such models triggers many questions to the developers about their exact ideas, although there was common agreement on the earlier informal drawings. By clarifying these issues in an early development phase, costly misunderstandings and repairs later on can be avoided.
The models are validated by extensive simulation and discussions with domain experts. Validation includes checking the completeness of the interfaces, the information that is available in each component, and whether the components can together collect enough information for making the right decisions. This concerns, for instance, sensor data and decisions where to store movement trajectories. By formally modeling different choices, the developers can really experience the consequences of different architectural decisions.
The analysis is particularly interesting when it comes to feature interaction. Because we have a single executable model for multiple scenarios, interferences can be identified early. For instance, in our model we had to make very explicit which manual movements are allowed during certain medical procedures. Moreover, the possibility to inject faults makes it easy to do experiments, for instance with graceful degradation strategies.

Tool requirements

The SHESim tool which we used initially to develop architectural model in POOSL is not very suitable for industrial usage. Important problem is that SHESim uses a Smalltalk license which does not allow industrial usage. Moreover, the current user interface requires many mouse clicks, there is no modern editor with content assist, there is no type checking, and modeling errors often only become visible by a run-time error during simulation. Still there is a need for a light-weight modeling tool which provides fast insight into requirements and early design decisions. It should fills a gap between expensive commercial modeling tools (like Matlab and Rhapsody) that require detailed modeling, often close to the level of code, and drawing tools (such as Visio and UML drawing tools) that do not allow simulation.
In the second phase of this use case, we experimented with a new Eclipse interface for POOSL which is being developed in WP 6.3. This new prototype turned out to be a promising step toward professional industrial tooling. Important new feature is the import mechanism which makes it possible to split a model into smaller parts and reuse components. But, as expected for such a prototype, there are a number of requirements for further tool improvement:

- Increase the possibilities to detect modeling errors as early as possible, including type checking and scoping
- The current Eclipse editor is purely textual; missing is the graphical view of SHESim, either as an editor or as a visualization possibility
- Possibilities for model debugging, such as breakpoints and inspection mechanisms
- Visualization of execution traces and Gantt charts in the Eclipse environment during simulation
- Support for the synchronized simulation of executable models running in different tools

Follow up (M24 / M36)

In the next iterations the model will be refined and extended to deal with a larger set of scenarios, especially concerning more complex movements and the safety mechanisms which are still missing in the current version. Aim is also to relate the architecture model to requirements models, e.g., concerning the priority of movements and image orientation. On a longer term, deployment of software on hardware and timing properties will be a challenging modeling topic. While working on these modeling issues, improved tool support will be evaluated.

3.2.13 Activity A13: Early verification of software design concepts using demonstrator

The reason for the new technology

Within Philips Healthcare, a new X-ray system will be developed which includes easier positioning of the positioning system and a simplified safety concept (bodyguard handling improvements).

Figure 39: X-ray stand and patient support table with 3D model.

The current (legacy) software for movement control and collision prevention contains a *3D model* responsible for prevention of table and C-arc collisions, *bodyguard software* responsible for collision prevention of patient and *force and current sensing software* responsible for collision detection. Decision making (restricting movements) is implemented locally in each module; which can lead on a system level to unnatural behavior when sensors interact with each other. Furthermore, currently, collision handling is implemented directly on the axis level, which implies that for each configuration change (e.g. more axes in table, different sensing modules); all higher-level software layers are affected. As the number of supported configurations is growing, this gives rise to complex specifications, implementations and exceptional behavior, which can lead to undesired long project release effort.

The stakeholders of the new technology

Stakeholders for the activity are system architects, software architects and software designers.

Approach

In a pre-study project, an initial (prototype) software library of the reference architecture positioning has been implemented. One of the new layers is supervision, responsible for simplified safety handling, which will gradually replace the current Movement Controller software. A kinematics layer is responsible for a separation of concerns between user interaction and the physical positioning system. The concept is well-known in many applications of robotics and 3D gaming. This new architecture enables modeling on several levels in the software.

Activities during last year

The following technical risks are mitigated as activities in this study:

- *Feasibility kinematics*
  - Rotation/angulation functional correct or issues known and solvable
  - Compute load in control or issues known and solvable
- *Feasibility supervision*
  - Proven to be safe on 3D / bodyguard collisions, or issues known and solvable
  - Compute load in control or issues known and solvable
- *Feasibility incremental migration path to new software architecture*

- Critical use case verified on target, new stand movements new combined with legacy table movement legacy, proven to be safe on 3D / bodyguard collisions, or issues known and solvable

- *Scalability/extensibility for future/external geometries*
  - Benchmark: combined movement with 7 axes
  - Compute load in control or issues known and solvable
  - Benchmark: add latency / jitter on table position (3D model) updates
  - proven to be safe on 3D / bodyguard collisions, or issues known and solvable

Results

Feasibility kinematics

The goal of kinematics is the separation of concerns between physical sensors and actuators (e.g. axes) and the higher-level safety and functional requirements. From user (and safety) point of view, use cases and safety requirements are specified in user perspective, so working in user understandable object movements (move table(x,y) instead of move axis x, move axis y) is natural.

The kinematics layer converts axis coordinates to object coordinates (and vice versa) between single-axis motion control and supervision. Furthermore, a conversion from object coordinates to room coordinates is needed for positioning of the table and stand in the examination room. The kinematics library is responsible for keeping track on axis limitations

See the figure below for an example for a multi-axis robot with the three different coordinate systems.

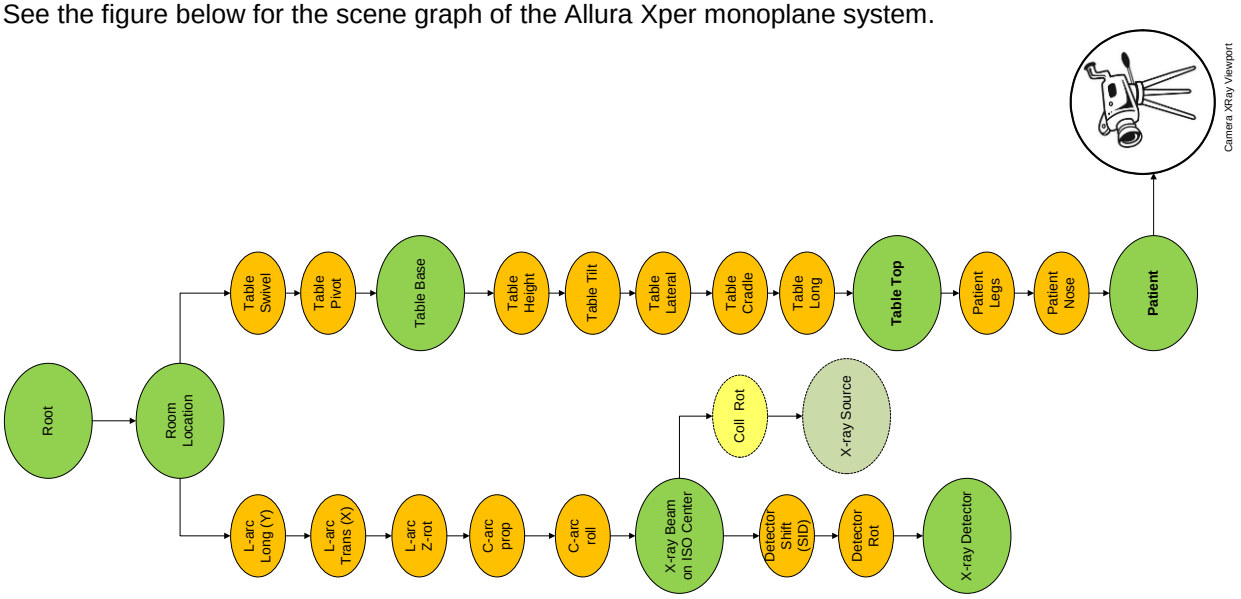
Room coordinates Object coordinates Axis coordinates

Design

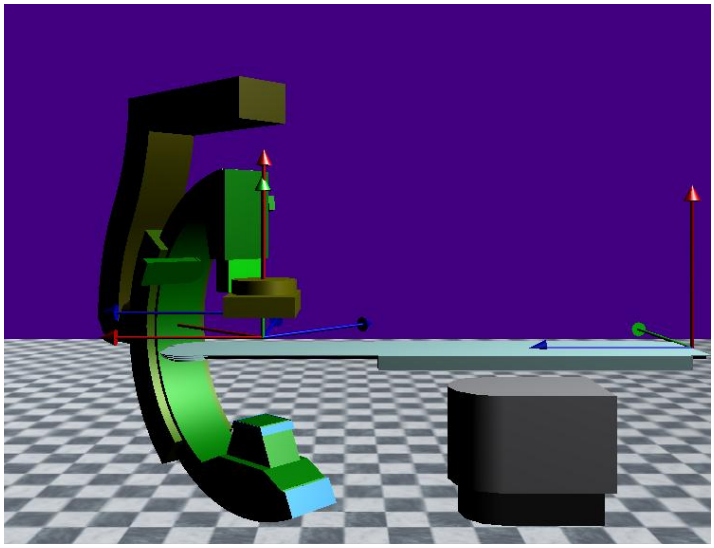
We have defined a world/room coordinate system (in the form of x,y,z directions + roll,pitch,yaw angles). A kinematic chain is then described in objects and can then be described in link and rotation points (analogous to scene graphs in 3D modeling/games). Each rotation point has its own coordinate system. Object movements are described in the objects parent coordinate system.

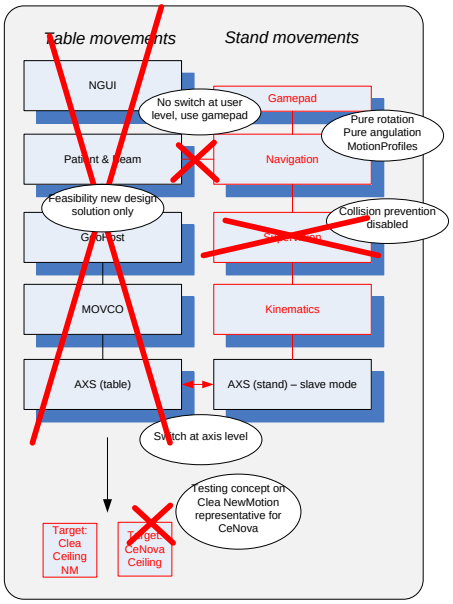
Alternatively, object movement requests could be described in the room coordinate system. The result would be additional complexity in generating new requested positions during the actual movement of the object (in the room).

See the figure below for the scene graph of the Allura Xper monoplane system.



The study has delivered a prototype supervision library, taken into account the 3D model objects.

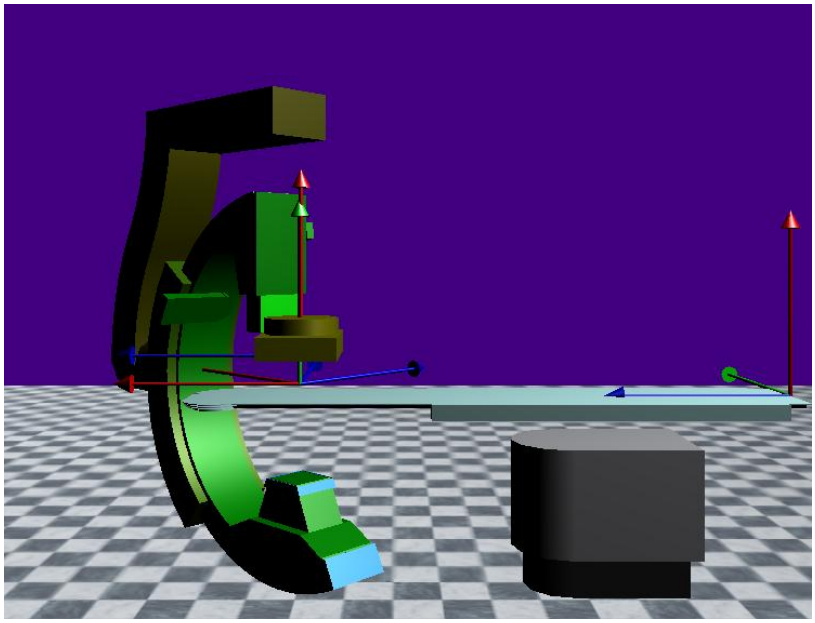


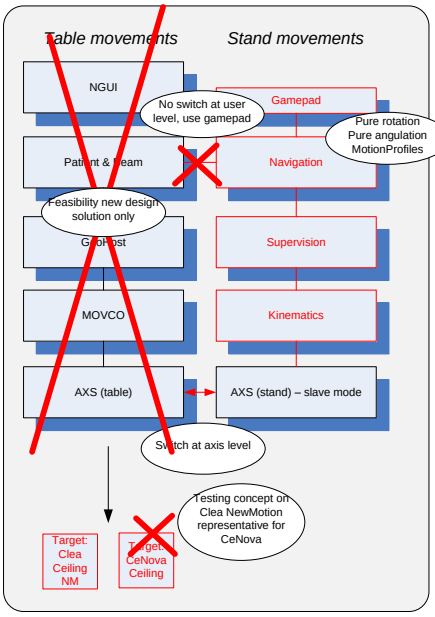


The kinematics use case consists of 27 tests, for 3 different Z-rot angles, for 3 different angulation angles, perform a rotation movement (-40,40). Repeat for angulation movements keeping rotation constant. Summarizing:
A: For three different stand Z-rot positions (1: -30, 2: -45, 3: -60 degrees)
B: Perform stand rotation (-40,40) around table for three different angulation angles (1: 0, 2: -40, 3: 40 degrees)
C: Perform stand angulation (-40,40) around table for three different rotation angles (1: 0, 2: -40, 3: 40 degrees)

- The path deviation with rotation and angulation movements is currently maximum 2 degrees, which is out of specification. Simulations confirm that with new stand limits, it will give us enough design freedom to decrease the path deviation to within specification. However, high speed movements at steep rotation/angulation angles will eventually become a problem, even with increased jerk/acceleration limits. To solve for this (fundamental) issue, the user-perceived speeds should be reduced or additional axis should be added to the movement (e.g. Z-rotation) Tests to confirm this are to be planned as future work.
- In the current solution, near singular points, the kinematics solver operates with limited free axis, which sometimes give rise to unpredictable behavior. From a requirements point of view, it is desired to avoid this behavior and stop the kinematic movement before reaching singular points or continue single-axis based. Detailed usability behavior is to be worked out in future work.
- Results confirm that the performance (without collision prevention) never traverse the 200 us (maximum).

Feasibility Supervision
The goal of supervision is the prevention of collisions of robotic objects moving in the examination room. The proposal we adopted is using a central mapping of the environment in a 3D model. The 3D model contains information of the surroundings to determine validity of movements while maintaining safe distance to nearby objects.
In current products, supervision acts directly on the (lower-level) axes layer, where navigation translates room coordinates to axes coordinates.
In the new situation, the kinematics layer abstracts away from geometry dependent axis information and supervision and navigation act upon more abstract positioning system objects such as table and stand.



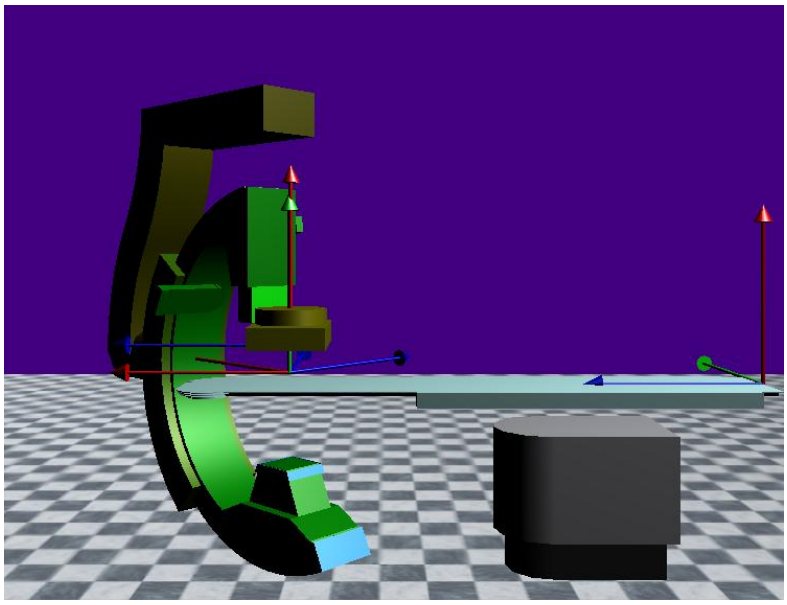


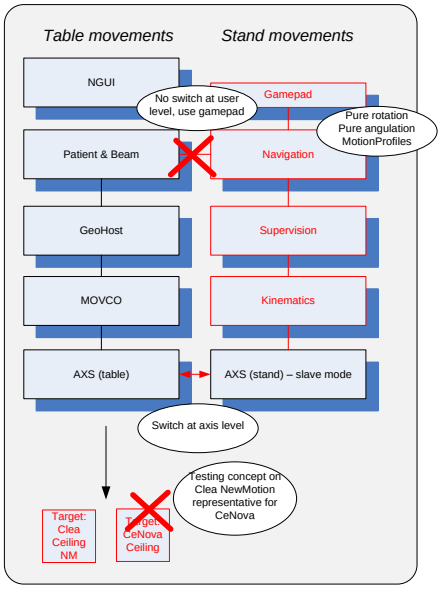
The supervision (collision prevention) use case consists of 9 tests, for 3 different Z-rot angles, for 3 different rotation angles, perform a angulation movement (-40,40). Summarizing:
A: For three different stand Z-rot positions (1: -30, 2: -45, 3: -60 degrees)
C: Perform stand angulation (-40,40) towards table for three different rotation angles (1: 0, 2: -40, 3: 40 degrees)

- Performance results confirm that the performance (with collision prevention) never traverse the 400 us (maximum).
- The path deviation is in the same orders as with the rotation/angulation movement (2 degrees maximum). For two test cases we see the path deviations increase to 6-8 degrees. Replay confirms that his is not due to the collision algorithm, but a result of a very steep projection in combination with a angulation movement. The current axis limits are unable to cope with the requested (patient-oriented) path.

Feasibility incremental migration path to new software architecture

To mitigate the risk of a big-bang (all or nothing) introduction of the new software architecture positioning, an incremental migration path has been defined with escape scenarios, in which an (intermediate) hybrid solution can be delivered to the market. The plan is such that the application leyer (Cockpit) and technical layer(kinematics, single axis IO) are gradually migrated to the reference software architecture positioning featuring Navigation, Supervision and Kinematics layers. Event-based control is chosen for the top level control layer (Navigation / Parameter handling), whereas loop based control is chosen for the lower-level layers (Supervision, Kinematics, Single Axis)





To verify the possibility to incrementally deliver the new software architecture, tests have been performed with patient-oriented stand movements in combination with legacy table movement software.
Test case:

- Move stand L-arm to in between position (30, 45, 60 degrees)
- Move stand angulation with tube towards table and concurrently move table towards tube (height or isotilt)
- Verify collision prevention of table towards stand and vice versa (with bodyguard disabled)
- Repeat procedure for different L-arm in between positions and different table movements

The above test cases has been verified on a target system with the legacy collision prevention model (CPCO) on for table movements, and the new collision prevention (XPoser) on stand movements. Manual testresults have shown the moving table and stand concurrently approaching each other to stop at 2 cm distance. This has also been verified by manual measurements.

Scalability/extendibility to new/external geometries
To prove the scalability of the kinematics solution for future geometries, a testcase (helicopter movement) has been created that solves for maximum number of axis (in this case 7 axis). While moving the table in tilted/cradled position, the C-arc in a (ideally: XY) configuration tries to follow with the detector the initial table position.
The overall performance is with a maximum of 600 us significantly higher (3x) in comparison to the rotation angulation movements (solving for 3 axis: 200 us max). Assuming the collision prevention to add an additional 200 us, the solution is sufficiently performing for future geometries.
To prove the extendability of the supervision (collision prevention) solution for external geometries, a testcase (ghost table movement) has been created that is representative for connecting and controlling an external table. As the current external tables are connected via a (shared) network connection, delay and jitter can occur on position updates and movement commands. This test introduces latency and jitter on the position updates, and checks if the supervision is robust to cope with additional latency on its input. The above concept is tested in simulation and proven to be safe for latencies up to 500 ms.

Tool requirements

In this study, we made extensive use of visualization tooling for both simulations and target testing.
There are a number of requirements for further tool improvement

- Store-playback functionality of use cases
- Import/Export of motion traces for analysis in external tooling (e.g. Matlab)

Follow up (M24 / M36)

As technical feasibility has been proven with this study, follow-up activities include the detailed software and interface design with respect to user needs. Moreover, this demonstrator enables the usage of models throughout the software layers. This method links to Engineering Methods VerifySoftwareArchitectureDesign and FormalizeUID.

3.2.14 Activity A14: Coupling requirements to verification test cases using HPQC

The reason for the new technology

Introduction

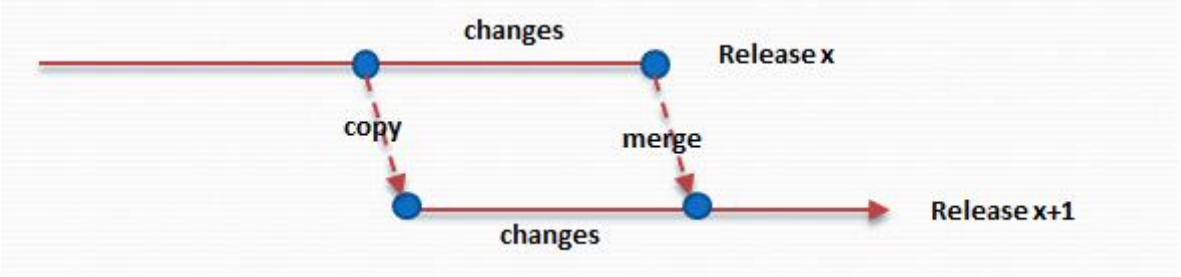
Traditionally products were developed according a sequential approach. A product is being developed and as soon as the product development reached the later stages of its development, a new project was initiated. Growing complexity of the systems lead to the first needs for a more parralel approach and with it camethe need for integration of subsystems into a system. With the introduction of software the whole approach needed redefinition, agile way of working, multiple software teams handling the increasing demand of software features. The last complexity factor is today's market where shorter time to market is needed to speed up innovation and handle price reducing demands. This all translates into variants, product families and configurations and configuration management.

The setup of the system can be devided into 3 major blocks:

- Mechanical parts (most important variability aspect for Philips Healthcare is hardware configuration of the system: 1 C-arm versus 2 C-arms, 1 large monitor versus 4 or 6 smaller monitors. operating or non-operating table etc.)
- Electrical parts (smallest need for variability management, component redesigns)
- Software parts (features, incremental development, etc.)

Besides identifying the need for variability management on components, development is done according the V-model. The V-model is a waterfall based approach. In today's world the focus is more and more on incremental development to handle complexity and rapidly changing needs of internal and external customers. This is especialy visible in software development.

As a prestudy to investigate the variability management needs a study has been done in one of the subsystem teams that creates software to support treatment with the use of the system. The study focuses mainly on handling requirements and test cases on changing features, developed in incremental and paralell approaches:



With the parallel development comes the need to merge activities from a branch to a baseline. In most of our process this remains a manual action which is, at best, supported by tooling. The application used to manage the requirements and test cases for this study is HP Application Lifecycle Management 11 with the following setup:

Situation:

- Working on 2 parallel releases on the same requirements and tests
- Requirements and tests reused from release x to release x+1
- Work is done in both projects at the same time
- When release x is done, changes of x should be merged into x+1

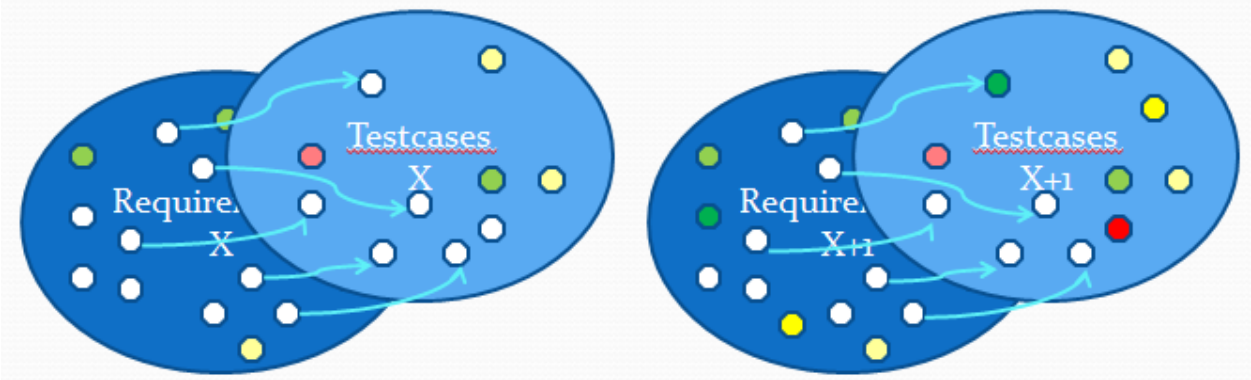
Challenge:

- How to manage/control the changes in requirements and testcases happening in parallel in both releases using tools

Complexity

In an end to end view a requirement is created for the product family. As soon as a specific release has been defined the requirements process will have to assign the requirements that are continued, changed, removed and added in relation to the defined predesessor or product family, basically creating a branch from the product family main line / baseline. For test cases the same activities need to be executed aswell as the traceability with its changes between the requirements and testcases (full coverage):

White = unchanged
Green = added
Yellow = changed
Red = removed



ALM options for parallel requirements
HP ALM11 is currently already in use for Test Management where one ALM project is created per system release. ALMs way of working is purely project based with no built in features for merging, branching, cross project version management or traceability. There are some features that help to exchange data (libraries and baselines) but these features are limited.

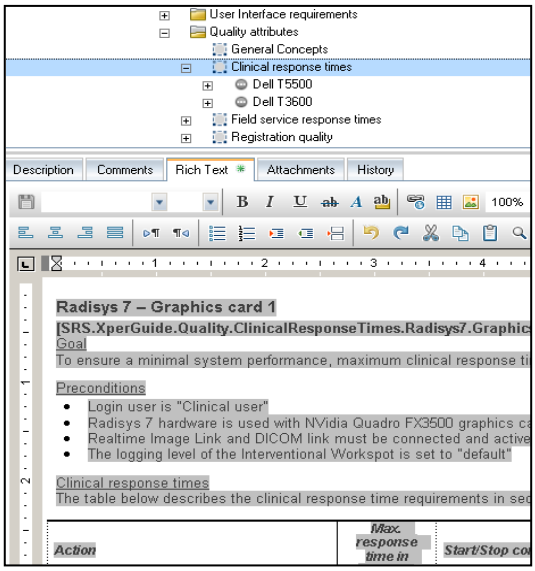
Activities during last year

The study is performed by the subsystem group that delivers supporting software for treatments. From a systems point of view the subsystem products need to be integrated into the system.

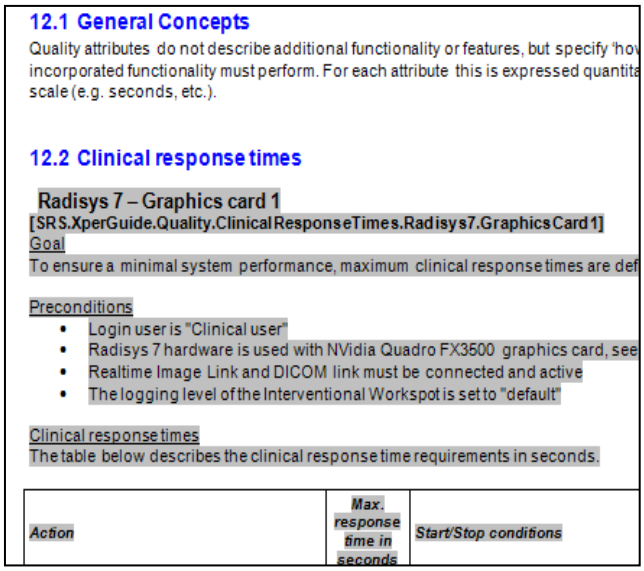
The following solutionshave beenstudied:

1. **Simple solution (similar output as could be created in Word, if worked in one document):**
   - Grey out text that is not applicable for the current release, and ungrey the specific parts in a next release and so on.
     - Pro:
       - very simple
       - known way of working (like current handling of Design History Files)
       - keep current and future requirements together in one record
       - easy manual merge by ‘ungreying’ during the next project
       - re-use of requirements across products is possible (if all products are in one repository database)
     - Con:
       - not using the potential of the tool
       - not possible to create/generate two versions of same document for both releases
       - does not support ‘deleting or changing for next project’
       - Can’t prepare traceability linking for future project

- Example: ALM11:

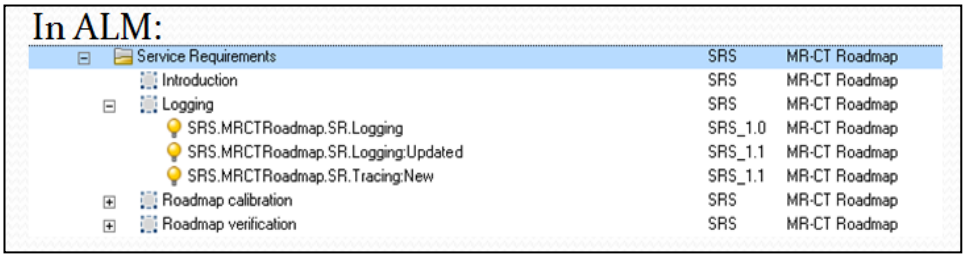


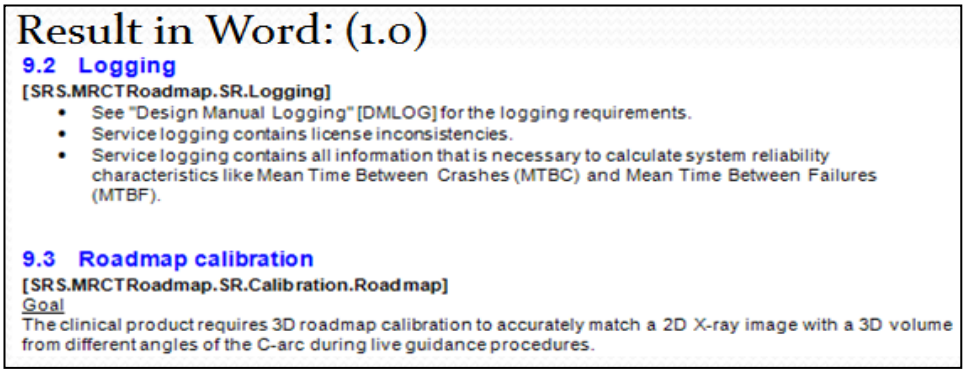
Word:

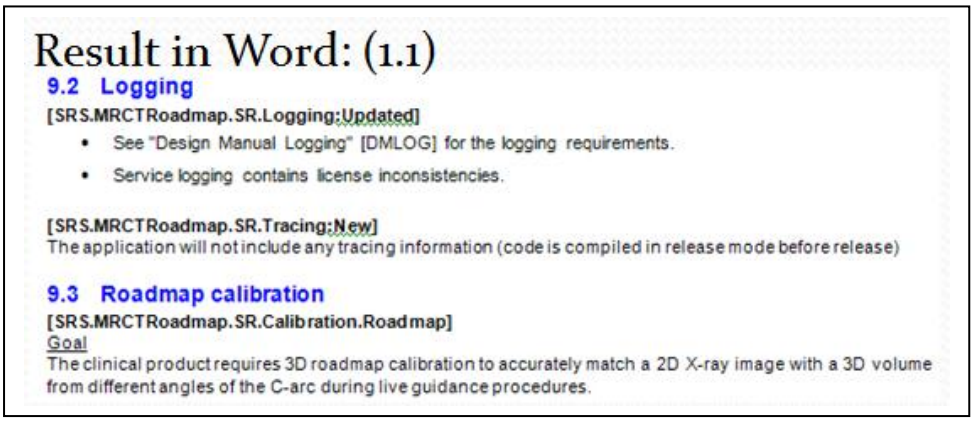


2. Use attributes to keep versions apart (2 records in 1Database)

- Change attributes at the appropriate time to reflect the 'current version'
  - Pro:
    - very simple
    - keep current and future requirements closely together in the tree view
    - easy manual merge by removing old record / updating attribute of new record
    - easy to create both versions of a document with requirements in the right place
    - re-use of requirements across products is possible (all products in one repository)
  - **Con:**
    - does not support 'deleting for next project'
    - updates in records for release x must be duplicated manually in record for x+1
    - the need to maintain attributes
- Example:

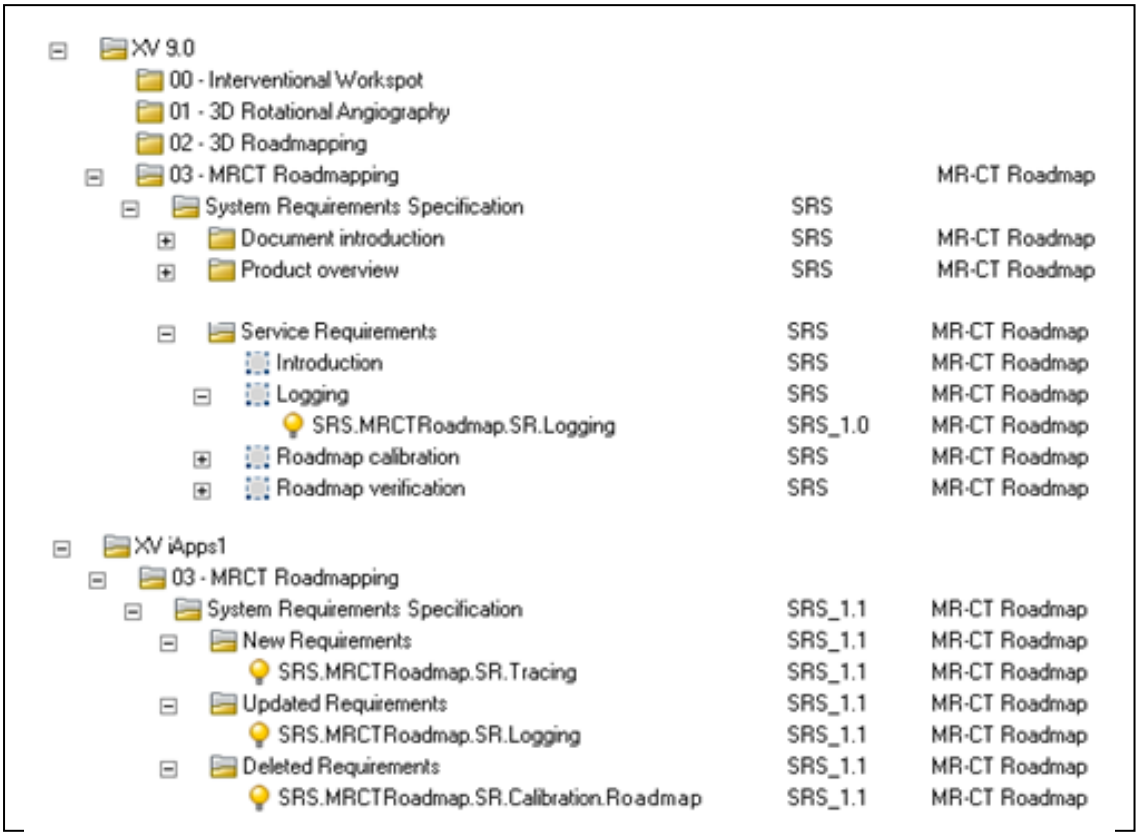






3. Use record attributes to keep versions apart, use separate folders for future requirements

- Change attributes of requirements at appropriate times to reflect the 'current version', and merge future requirements into the tree later
  - **Pro:**
    - very simple
    - easy manual merge by removing old records and merging updated record into tree
    - easy to create both versions of document, with new/updated reqs in one chapter and the end.
    - supports easy deletion of future requirements
    - proven concept, used for testcases in XV881 vs XV9
    - re-use of requirements across products is possible (all products in one repository)
  - **Con:**
    - future requirement version is not at correct (final) location when generating the word doc;
    - not a problem for entire new products, but not so nice for changing legacy products. (changed reqs are at end of doc, not at correct location)
- Example:

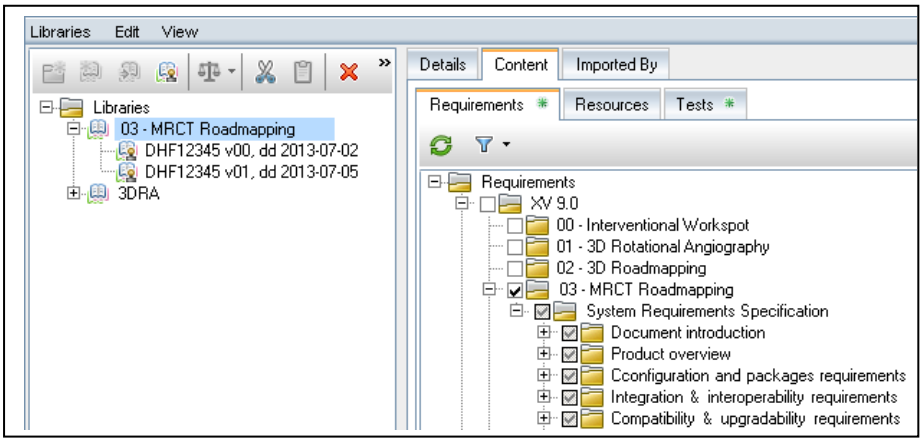


4. Use Library / baseline mechanism from ALM in combination with multiple databases

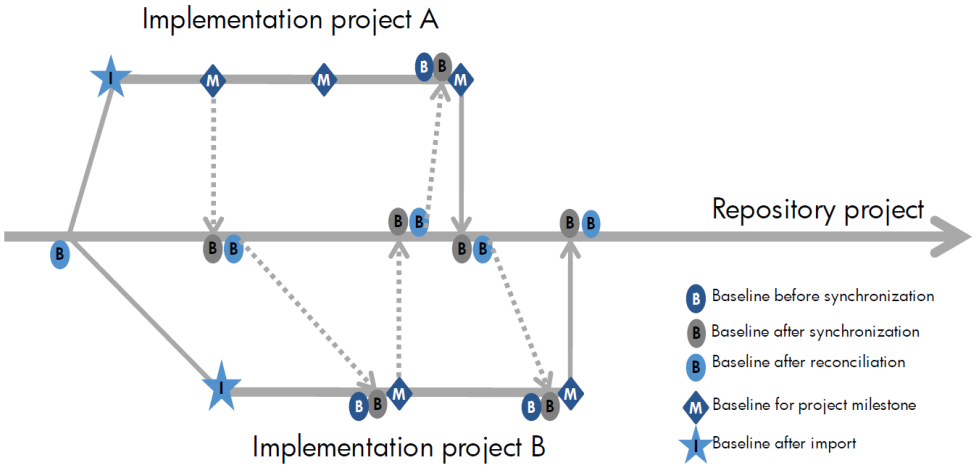
- Set up 3 databases, one repository, one release x, one for x+1
- Create a library (definition of a filter on what to baseline) of a particular product, in the repository database
- Create a baseline of the library in the repository databasewhen starting a release, and export the baseline to the project database
- Edit requirements in either the repository (if change is generic) or in the release database.
- Decide on when to merge back to repository from release x, or x+1

    - **Pro:**
        - Solution proposed by HP for complex parallelism
        - Explicit manipulation of baselines in multiple databases, but not by the requirements owner role
        - Easy to create both versions of a document (in two databases)
        - Supports easy deletion of future requirements
        - Re-use of requirements across products is possible (all products in repository)

    - **Con:**
        - High maintenance on baselines, merges, but also on reports, queries, filters etc
        - At least three parallel databases for everybody to work in
        - Need agreements between projects when baselinesare made, reconciled, merged etc.
        - Need special user access rights to merge
        - Complex



- Define libraries on product basis
  - select Reqs from tree, or via a selection filter
- Keep Reqs versions and Tests versions together
- Name baselines as DHF numbers + versions, as in example above
- Allows comparisons between baselines
- Allows generation of documents based on baselines
- Does NOT allow branching/merging in same database
- Is intended to export 'final' baseline into separate database, for release purposes (MR way of working)
  (i.e. Export Baseline at RfV, build up testevidence)
- Requires double maintenance on requirements and testcases after RfV, if updates are needed in Release database, or a merge back to repository (not tested yet)



Source: HP

5. Use database per product / per release

- Set up product databases, one for each release of product
  - Pro:
    - Each product / release in a separate database
    - Small requirements trees
  - Con:
    - Each team works in multiple databases (each team has more than 1 product, and maybe more than 1 release)
    - No re-use of requirements between products possible
    - High maintenance on reports, queries, filters, TTM, because you need to do that in every DB separately.
    - Multiple databases for everybody to work in
    - Basically makes problem not go away… (parallelism is not addressed, but smaller only databases)

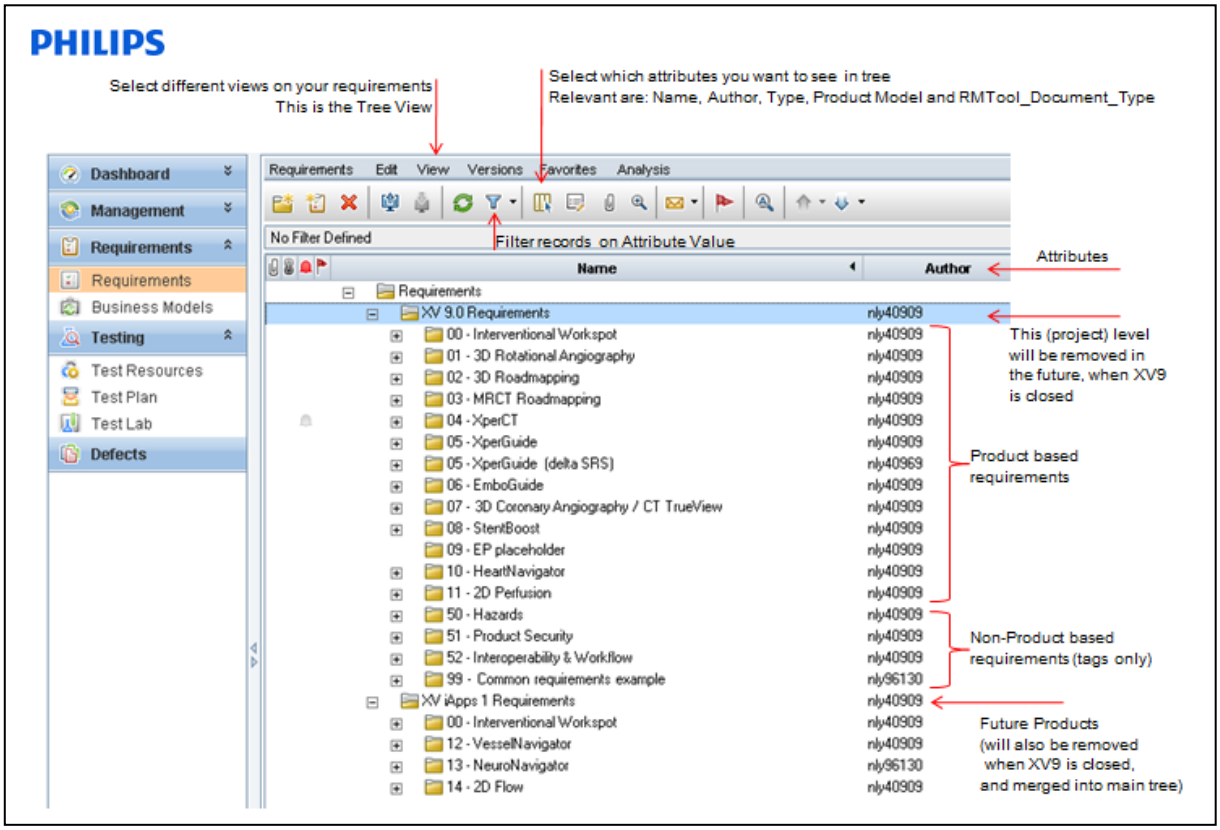
Results

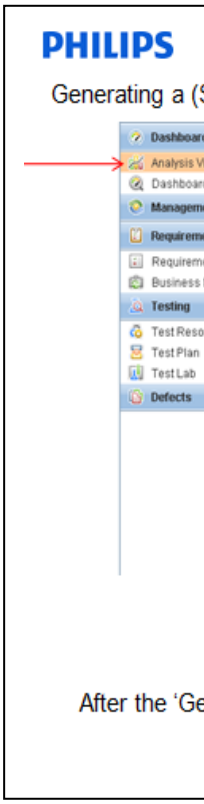
The study has resulted in a training tailored to the needs of the subsystem group.

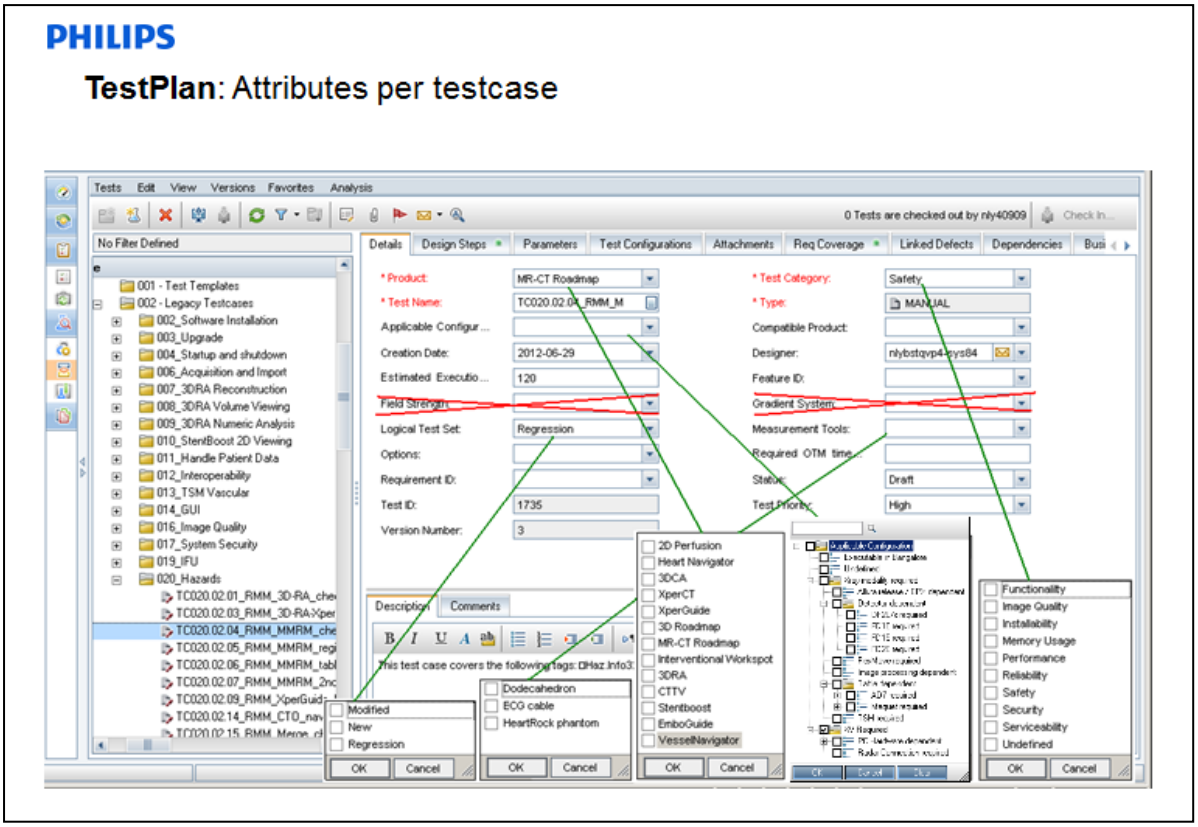
In the training the following subjects are adressed:

- Document Way of Working in ALM for subsystem Projects
- Align WoW between subsystem projects
- Help with quick introduction for new team members
- Help with quick introduction for developers

Examples of training issues are visible in the following sheets.


PHILIPS
Select different views on your requirements
This is the Tree View
Select which attributes you want to see in tree
Relevant are: Name, Author, Type, Product Model and RMTool_Document_Type
Dashboard
Management
Requirements
Requirements
Business Models
Testing
Test Resources
Test Plan
Test Lab
Defects
Requirements Edit View Versions Favorites Analysis
No Filter Defined
Filter records on Attribute Value
Name
Author
Attributes
Requirements
XV 9.0 Requirements nly40909
00 - Interventional Workspot nly40909
01 - 3D Rotational Angiography nly40909
02 - 3D Roadmapping nly40909
03 - MRCT Roadmapping nly40909
04 - XperCT nly40909
05 - XperGuide nly40909
05 - XperGuide (delta SRS) nly40969
06 - EmboGuide nly40909
07 - 3D Coronary Angiography / CT TrueView nly40909
08 - StentBoost nly40909
09 - EP placeholder nly40909
10 - HeartNavigator nly40909
11 - 2D Perfusion nly40909
50 - Hazards nly40909
51 - Product Security nly40909
52 - Interoperability & Workflow nly40909
99 - Common requirements example nly96130
XV iApps 1 Requirements nly40909
00 - Interventional Workspot nly40909
12 - VesselNavigator nly40909
13 - NeuroNavigator nly96130
14 - 2D Flow nly40909
This (project) level will be removed in the future, when XV9 is closed
Product based requirements
Non-Product based requirements (tags only)
Future Products (will also be removed when XV9 is closed, and merged into main tree)


PHILIPS
Generating a (S
After the 'Ge



Conclusions

After carefully piloting each situation the conclusion is that several solutions are available, but there is not a single solution that is without challenges. Tooling wise ALM11 does not have a major contribution in handling paralell projects, merging and cross project reporting functionality. All in all this leads to the need for manual actions and as with all manual actions comes error proneness and maintenance activities.

The study learned us that working with product families or variants is indeed a challenge where we currently do not have the right solutions in place. The study was done on a subsystem level in order to have a set scope and limited group of users. The posed challenges for the subsystem will have a larger impact on system level as there are more requirements (15k + for the product family) and test cases and a lot more people working in different approaches.

For the coming Crystal period a clear definition and the impact of product families and variability management needs to be defined (especially on system level). Also direct impact of variability management needs to be determined on the engineering method verify requirement including the scheduled pilot projects.

Tool requirements

Tool requirements for variability management will be available soon after M12.

Follow up (M24 / M36)

- Definition of product families and variability management (especially on system level).
- Definition of IOS adapter for HPQC

3.2.15 Activity A15: M9 demonstrator Caliber – HPQC – IBM RQM

For a description of this demo, see D401_021.

3.2.16 Activity A16: M12 Demonstrator: Integrated demo WP4.1 + WP4.3

For a description of this demo, see D403_901.

3.3 Engineering workflow at M12

Improvements achieved in the engineering workflow at M12:

- User needs concerning patient examination table and stand positions and movements can be validated by means of simulations. 3D- models are used as a “3D specification”, which enables medical end users, applications and marketing specialists to evaluate the requirements specification. (Activity A3, A4, A7, A8)
- Risks of unforeseen effects of system configurations on usability (positions, movements, room set up) can be reduced significantly by means of a early 2D simulation;
- Specification quality can be increased significantly by formalizing and analyzing procedures;
- A start has been made to investigate interoperability of tools.

Although these improvements are a step in the right direction:

- Most of the engineering methods, models and tools are rather dedicated. As a consequence, a large number of models and tools are needed to cover the workflow improvements.
- These tools are hardly interoperable, tools and models are not reusable. Result: the use of tools is labour intensive; models of the same subject can hardly be reused and are created time and time again.

Figure 40 depicts how the 16 activities of M12 4.1 map on general development process in a critical system engineering environment.

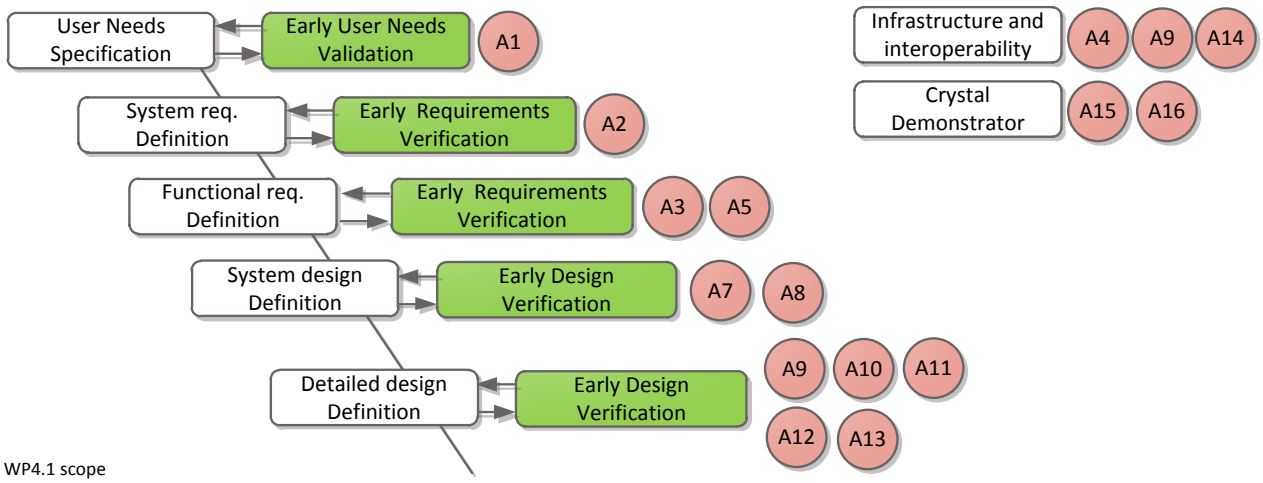


Figure 40: Mapping of activities on development process

3.4 Envisioned Engineering Workflow

As a next step towards the Envisioned Engineering Workflow, we aim at multidisciplinary approaches with reuse of models throughput the entire development process.

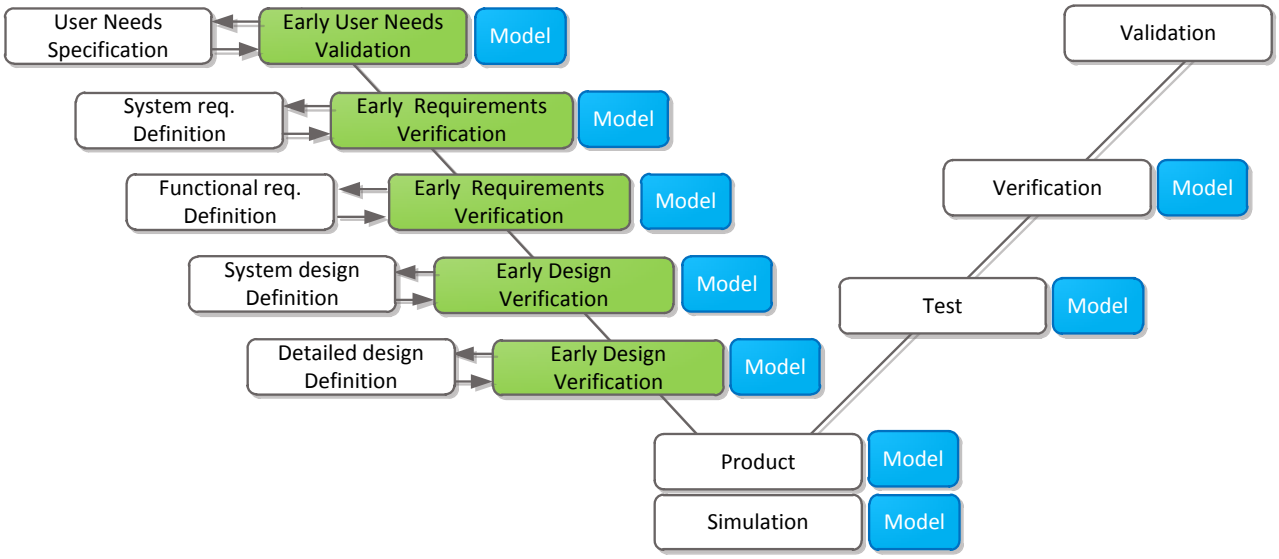


Figure 41: Envisioned Engineering Workflow

In Figure 41 the development flow is visualized, targeting at reuse of models across all development stages:

- Early User Needs validation: Provide means to interchange 3D visualization with visualization and simulation tools currently used for early design verification, and enrich visualization tools with more realistic visual-reality rendering to help clinical marketing and application with validation of new product features.
- Early Requirements Verification: Make DSL integrated part of SEE, Create an automatic connection between requirements, design and code using DSLs
- Early Design Verification: Create demonstrators using models integrated with the physical test system. Generate code based on the identified models, to speedup process of creating demonstrators and gaining feedback.
- Product creation, simulation, test and verification: Reuse of selected models used in early verification development stages, such that product development is no longer a separated task from product de-risking activities.

3.5 Engineering Methods

Engineering Methods provide a technical description of activities and scenarios which make up the overall use case from an end user perspective. They describe the general problem and workflow and the envisioned solutions. The Engineering Methods are defined by the Use Case Owners.

WP4.1 is limited to the following set of Engineering Methods:

- *UC401_Formalize_UID*
- *UC401_Verify _Software_Architecture_Design*
- *UC401_Verify_Requirements*

The relation between the 4.1 Engineering Methods and the Crystal Improvement Themes[5] is visualized in the following table.

Engineering Method	Crystal Improvement Themes			
	Individual models and the modelling architecture	Model engineering infrastructure	Optimizing the engineering workflow	Institutionalizing changes to the Systems Engineering Environment
UC401_Formalize_UID	X	X	X	X
UC401_Verify _Software_Architecture_Design	X			X
UC401_Verify_Requirements				X

Table 2: Relation between WP4.1 Engineering Methods and Crystal Improvement Themes[3]

The paragraphs below provide a high-level overview on the Engineering Methods. More detailed information is available in chapter Appendix A (Appendix A) and “Technical Management” section “Engineering Methods” in the Crystal project archive.

[5] For more information see section 2.3: Challenges at M0.

3.5.1 Engineering Methods WP4.1: an overview

Engineering Method	Description	Tooling	Stakeholders
UC401_Formalize_UID	Scripting User Interaction Design (UID) in rapid-prototyping environment and visualizing UID in a 3D rendering animation Formalize the natural language specification in a DSL of user interaction design patient and beam positioning, with the aim to bridge the gap between system design and detailed design/implementation.	- Eclipse Xtend/Xtext (DSL), - DoorsNG (Requirements Management) - POOSL (rapid-prototyping), - XPOSER (3D rendering), - Nobi-VR (virtual reality system)	- Clinical application and marketing (get the right feedback / pre-validation on UID to system design) - System design (get an agreed UID spec), - Software architect and designers (translate UID into software design)
UC401_Verify _Software_Architecture_ Design	Pre-verification of software architecture and design on interface compliance for new or changed user interaction scenarios, without the need to actual implement the feature	- Visio (UML drawing tool), - POOSL (rapid-prototyping)	- Software architect and designers (early verification of design)
UC401_Verify_Requirements	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.		- Clinical application and marketing (get the right feedback / pre-validation on UID to system design) - System design (get an agreed UID spec), - Software architect and designers (early verification of requirements)

Chapter Appendix B provides a high-level overview of the Engineering Method. More detailed information is available in the “Technical Management” section “Engineering Methods” in the Crystal project archive.

3.5.2 Relation between Crystal activities WP4.1 and Engineering Methods WP4.1.

Nr		*UC401_Formalize_UID*	*UC401_Verify _Software_Architecture_Design*	*UC401_Verify_Requirements*
A1	Activity A1: Early concept validation of mechatronics using 3D virtual reality viewer	X		
A2	Activity A2: Early visual verification of system requirements using 2D viewer	X		
A3	Activity A3: Functional Requirements Analyzing and Formalization using DSL	X		
A5	Activity A5: Early visual verification of formal requirements in DSL using 3D viewer	X		
A6	Activity A6: Couple DSL to requirements management tooling using OSLC	X	X	X
A7	Activity A7: Early verification of system design concepts using 3D viewer	X		
A8	Activity A8: Early verification of system design concepts using demonstrator		X	
A10	Activity A10: Early verification of mechatronics design concepts using Matlab and 3D viewer	X		
A11	Activity A11: Early verification of mechatronics design concepts using demonstrator			
A12	Activity A12: Early verification of software design concepts using POOSL		X	
A13	Activity A13: Early verification of software design concepts using demonstrator		X	

Chapter Appendix B provides a high-level overview of the Engineering Method.. More detailed information is available in the “Technical Management” section “Engineering Methods” in the Crystal project archive.

3.5.3 Engineering Method and IOS

For the engineering method Verify Requirements, we have analyzed the need for IOS services based on the steps in the Engineering Method. In a meeting with an IOS specialist and the brick-owners IBM and PTC we came up with a list of IOS services that are needed to build the interoperability between the tools.This is a direct way to extract IOS services out of the Engineering Method.
These IOS Services will then be handed over to WP6.1 for consolidation.

As a results of the meeting, also we gained new insights in the EngineeringMethod VerifyRequirements; this EM was updated to provide a complete overview of the steps and the interaction between the tools.

An example of the EM / IOS matrix can be found in Appendix B, EM VerifyRequirements.

4 Building Systems Engineering Environment (SEE)

4.1 Introduction

The basic view behind a System Engineering Environment supporting Model Driven Developments is based on the following points:

- The selection of the requirements language determines the tools to create the model;
- The output of the modelling activities can be used:
  - in a product;
  - In a visualisation tool to show the contents to the other stakeholders: in a demonstration, 3D/2D presentation of the system and its behaviour, etc.
  - As an input for test cases.
- The actual creation of a model shall be under version/configuration control, by a suited requirements management tool.
- For interoperability purposes the models shall be provided of annotation mechanism.

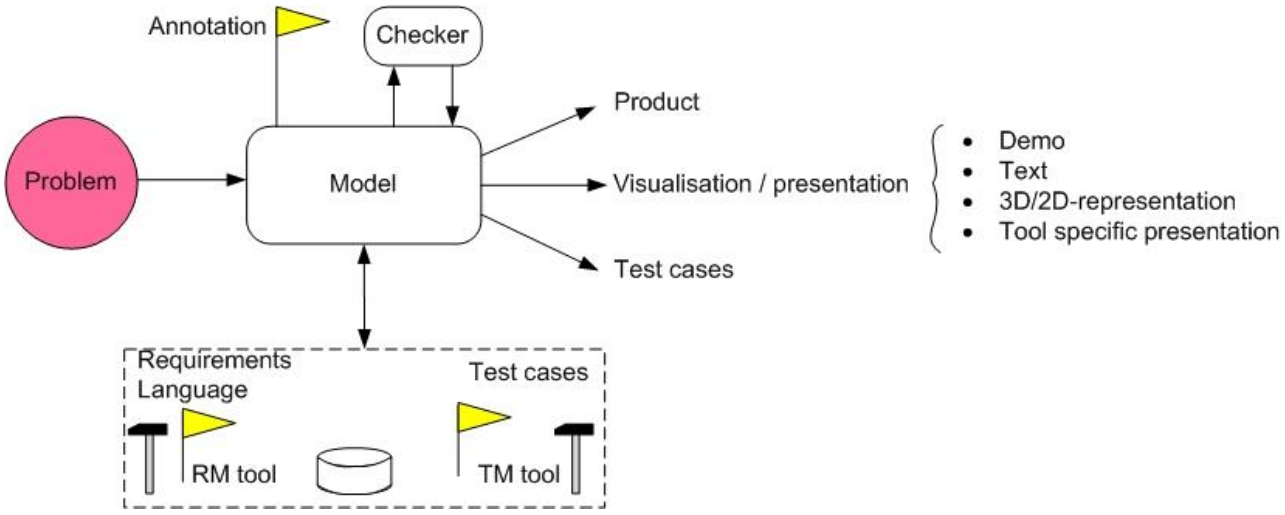


Figure 42: basic concept of a model in its system engineering environment.

4.2 SEE at M0

The System Engineering Environment at M0 shows a number of standalone environments for modelling, simulation and visualisation:

- Simulink and Matlab;
- UML and Rhapsody;
- Dedicated modelling languages and tools for visualisation.

Requirement specifications are mainly based on natural languages, processed in Word, managed in Agile.

Code (generated by means of Visual Studio) is stored and managed in Clearcase, test cases in ClearQuest.

Visualisation of requirements and/or design aspects is performed in dedicated standalone simulation environments.

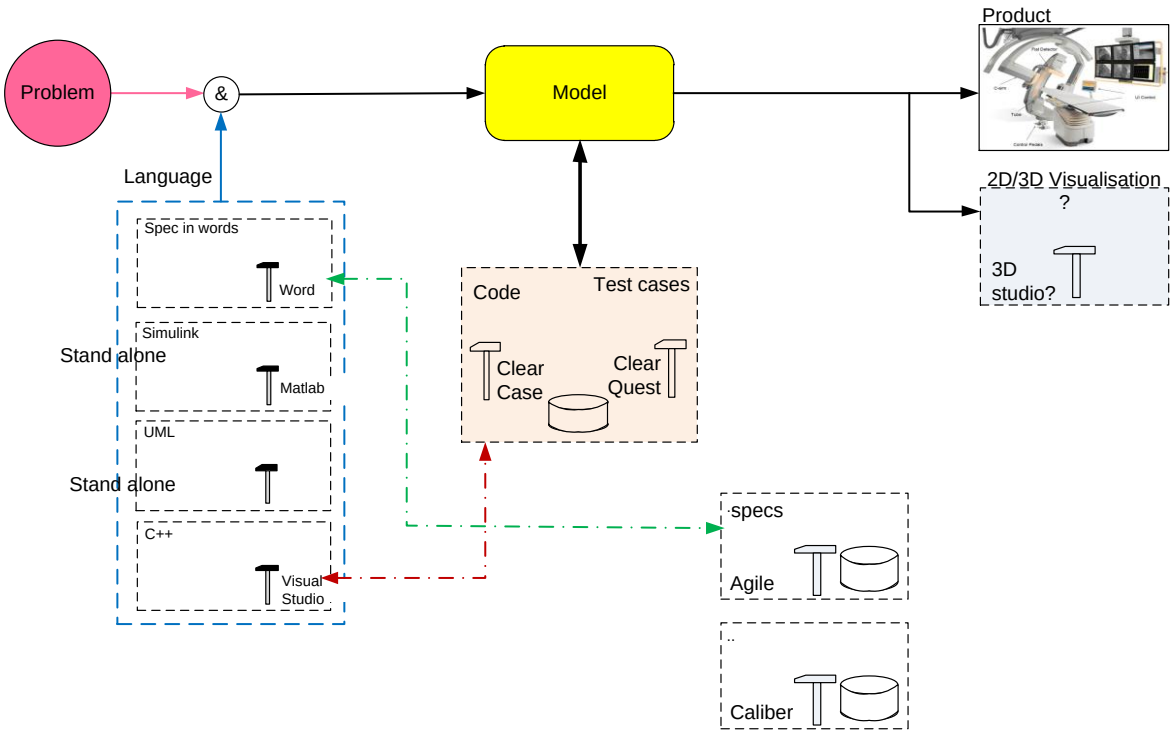


Figure 43: System Engineering Environment at M0 shows a number of standalone environments for modelling, simulation and visualisation.

4.3 SEE Initiatives started

The engineering workflow related activities are listed in the following overview:

Activity	Section
Activity A3: Functional Requirements Analyzing and Formalization using DSL	3.3
Activity A4: Infrastructure to early visual verification visualize using 3D virtual reality viewer	3.4
Activity A6: Couple DSL to requirements management tooling using OSLC	3.6
Activity A9: Early verification of mechatronics design concepts using Matlab	3.9
Activity A12: Early verification of software design concepts using POOSL	3.12
Activity A14: Coupling requirements to verification test cases using HPQC	3.14

4.4 SEE at M12

The figure below provides an overview of the Systems Engineering Environment that is currently in place at M12.

The SEE at M12 is characterised by the following points:

- Introduction of Simulink/MatLab, UML/Rhapsody, C++/VisualStudio, in an integrated environment with ClearCase.
- Introduction of Caliber for requirements management in an integrated environment with Agile and Word as an editor for specifications in natural language.
- Experiment with the introduction of DSL with XText/Eclipse visualisation (activity A3, see 3.2.3).
- Infrastructure to early visual verification visualize using 3D virtual reality viewer (activity A4, see 3.2.4).
- Introduction of Simulink/Matlab models with Xposer visualisation for analysis of design concepts (see activity A6, see 3.2.6).
- Introduction of an early verification of mechatronics design concepts using Matlab and 3D viewer (activity A10, see 3.2.10)
- Experiment with an early verification of software design concepts using POOSL (activity A12, see 3.2.12).
- Coupling requirements to verification test cases using HPQC (activity A14, see 3.2.14).

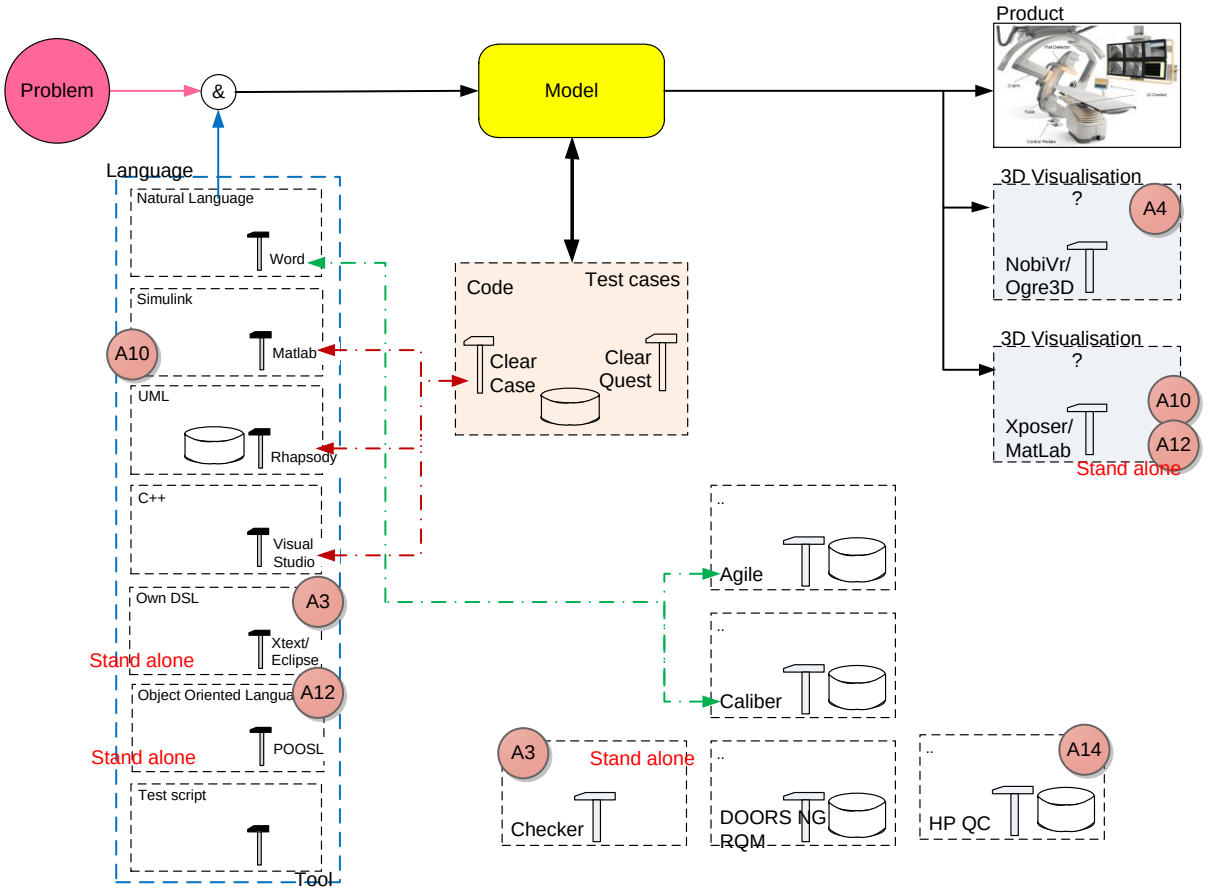


Figure 44 Overview of the Systems Engineering Environment at M12

See chapter 3.5 for a short introduction on the individual engineering tools (a.k.a. Brick) mentioned in the Systems Engineering Environment at M0 and M12, and the associated engineering artefacts they process.

5 Demonstrator descriptions

For a description of the common demonstrator for WP4.1 and WP4.3 see *Use Case Development Report UC403 Motion control of patient table and X-ray beam positioning.*

6 Conclusion and way ahead

6.1 Evaluation

As an answer to an increased design complexity due to higher demands on flexibility in the clinical room layout together with an increased variability triggered by efforts to adapt the same product platform for a broader audience, we have investigated the use of modelling in WP4.1.

At the same time, early verification of system concepts and reuse of modelling effort in the engineering flow is needed for creating acceptable time-to-market for safety critical system engineering products.

In the first twelve months of the CRYSTAL project, activities A1, A2, A3, A5, A7, A8, A9, A10, A11, A12 and A13 cover individual models in the ecosystem architecture.

Models are recognized as a means to counter complexity by raising the level of abstraction as requirements aid by defining the desired product behaviour (e.g. behaviour models).

- Activity A1 and A7 revealed that 3D visualization a good way to discuss and gain early feedback from clinical users on new concepts and requirements and as design aid by defining the actual product behaviour (e.g. architectural / structural models)
- Activity A3 with DSLs and model checkers revealed several inconsistencies in the current user interaction requirements specification that otherwise would be found late in the project at a high cost
- Activity A10 and A9 revealed that modeling is a requisite for manage the complexity of 3D multi-axis patient-oriented movement concepts as verification aid by predicting product behaviour (e.g. emulation or simulation models)
- In Activity A11, A8 and A13, demonstrators are created on a physical target system. Although the feedback is of high value as validation aid by providing early clinical feedback on the product behaviour, the cost of creating such demonstrators is high, and puts an additional load on critical resources in the project.

We conclude from the activities that our current way of working still lacks institutionalized (multi-disciplinary) reuse where the software group is mainly supporting the mechatronics and system disciplines.

With respect to tooling, individual modeling tooling is very useful in the development process to decrease time to market. However, without reuse of models and interoperable tooling, for each new clinical feature or technology, the R&D activities are started from scratch.

Activities A4, A5, A6 and A14 are related to modelling of an engineering infrastructure with the aim to bridge the gap between the individual modelling creating reuse. As a second outcome, the activities aim at creating an overview of the relations between the models in the different levels of the V-model to optimize the engineering workflow.

The work and demonstrator described in Activity A14 and A15 has proven that interoperability on the top (requirements-verification) level of the V-model has a high added value. However, the manual steps need further improvement including investigation of variability aspects.

The demonstrator of Activity A16 has made a first approach on integrating and reusing individual models throughout the entire left-side of the V-model. This activity has incorporated several activities from both WP4.1 as WP4.3. It proves that the conceptual approach chosen in CRYSTAL has added value and sets a clear direction for the future.

6.2 Planned future work

In the M12-M36 timeframe, we plan to extend the current approach as explained in the document, increasing the amount of tool interoperability for more complex models. In Figure 45: Future work on tool integration and reuse of models, an outlook is given how this would look like. Central in this approach is the Domain-Specific Language (DSL) component, which plays a central role to achieve the desired reuse. The tooling interoperability should focus on linking the languages in which the models are described. Therefore, the tools should support annotations in each individual language. Note that the tools that support the storage (e.g. IBM Rational ClearCase) are only a storage means and do not need domain-specific annotation.

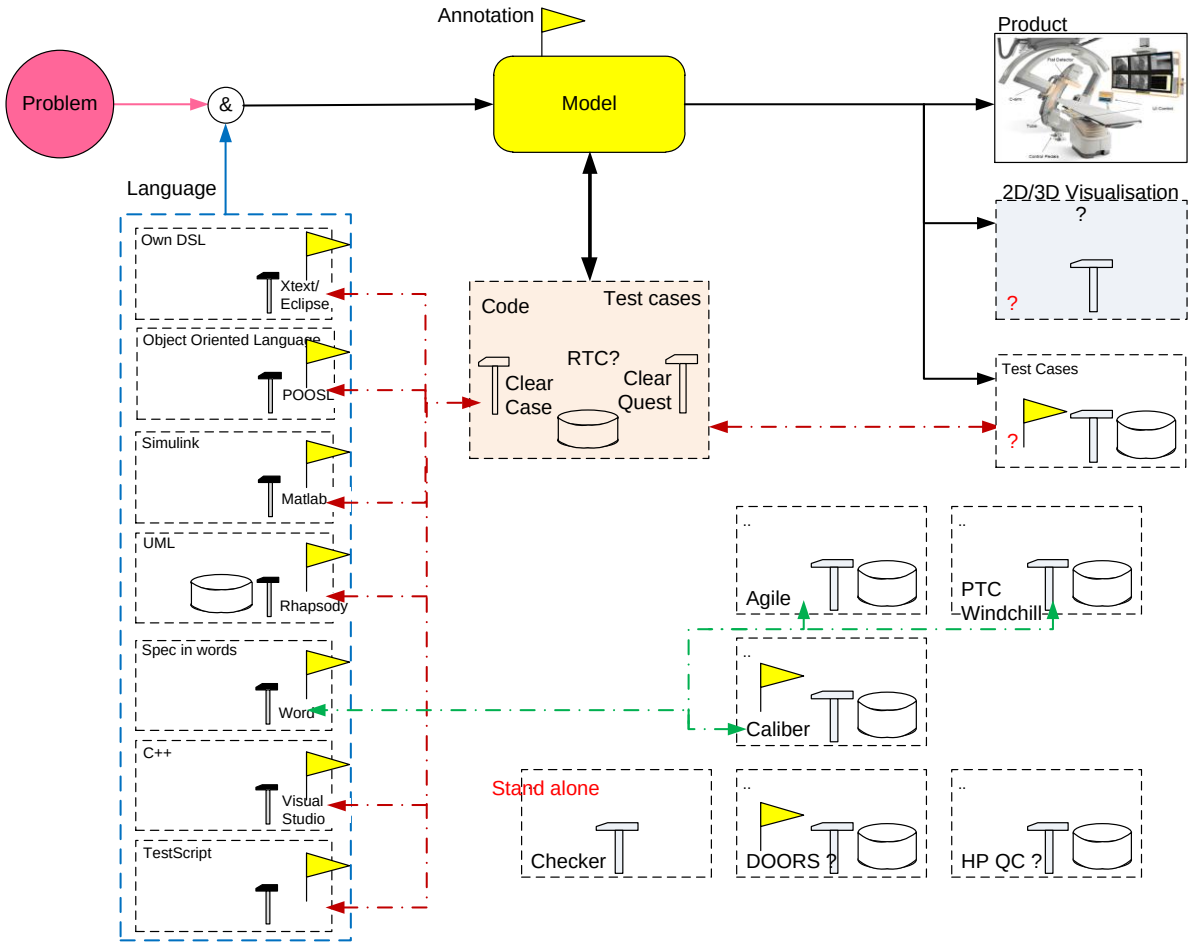


Figure 45: Future work on tool integration and reuse of models

7 References

[1] Albers, J.: Use Case 4.1 Medical Procedures in an Interventional X-ray System, D401.010 (2014). Update see chapter **Error! Reference source not found.**.

[2] Theelen, B.D., Florescu, O., Geilen, M., Huang, J., van der Putten, P.H.A., Voeten, J.: Software/hardware engineering with the parallel object-oriented specification language. In: Proceedings of MEMOCODE 2007, pp. 139–148. IEEE (2007)

[3] Li, L., Hooman, J., Voeten, J.: Connecting technical and non-technical views of system architectures. In: Proceedings of CPSCom 2010, pp. 592–599 (2010)

[4] J. Hooman, A.J. Mooij, and H. van Wezep. Early fault detection in industry using models at various abstraction levels. In Proceedings 9th International Conference on Integrated Formal Methods (iFM 2012), pp. 268-282. LNCS 7321, Springer-Verlag (2012)

[5] A.J. Mooij, J. Hooman, and R. Albers. Early fault detection using design models for collision prevention in medical equipment. In Foundations of Health Information Engineering and Systems (FHIES 2013), pp. 170-187. LNCS 8315, Springer-Verlag (2014)

[6] Blender http://www.blender.org/

[7] Ogre http://www.ogre3d.org/

[8] Anim8or http://anim8or.com and http://en.wikipedia.org/wiki/Anim8or

Appendix A High level description of use case and context

A.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 46 (left) outlines this V-model while Figure 46 (right) maps this onto the documentation. Figures are borrowed from internet sources and Mouz et. al. (1996,2000).

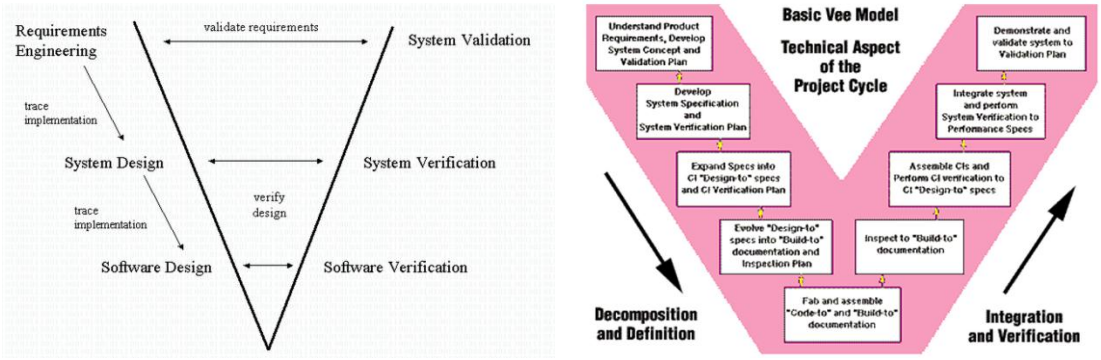


Figure 46: the V-model showing the process (left) and the documentation (right).

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.**

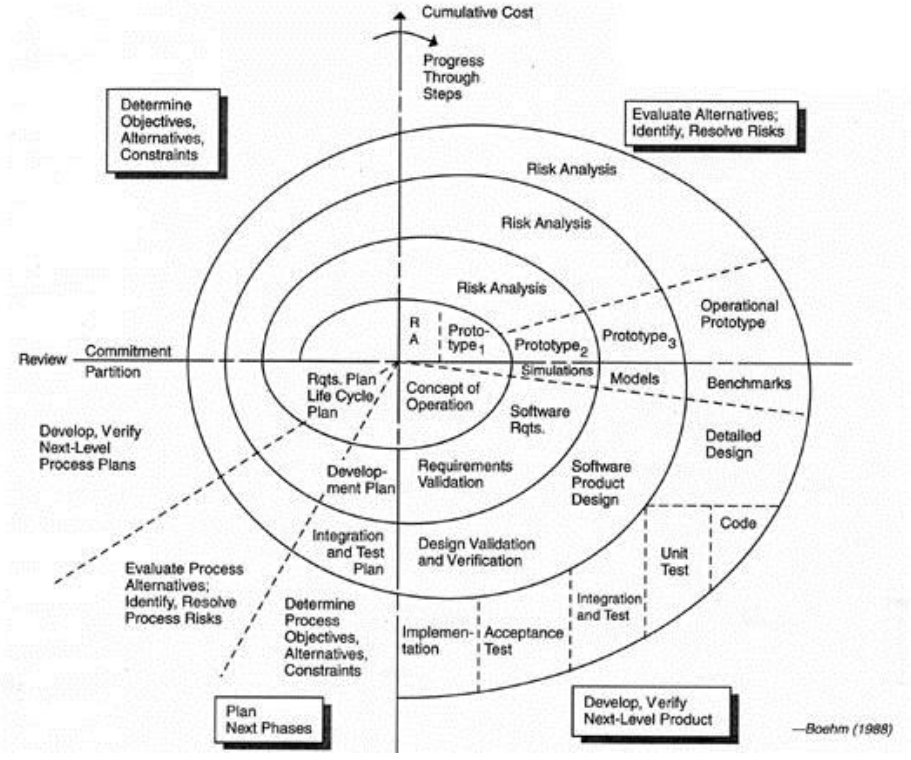


Figure 47 Spiral Development [Boehm 1988]

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.

A.2 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.

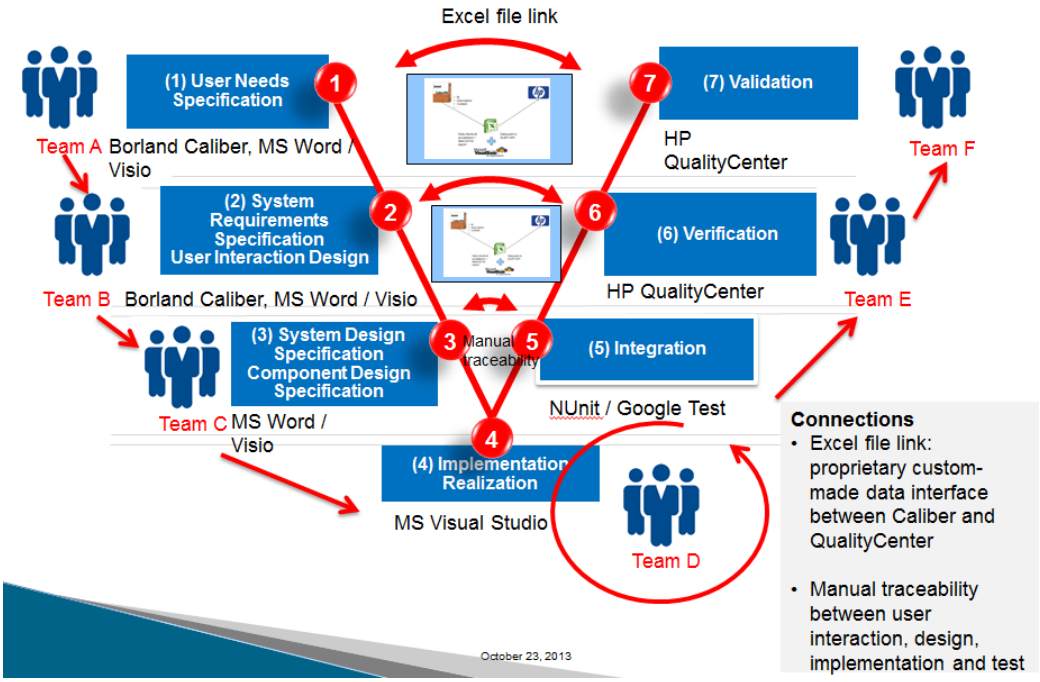


Figure 48 Current development process

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.

	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.

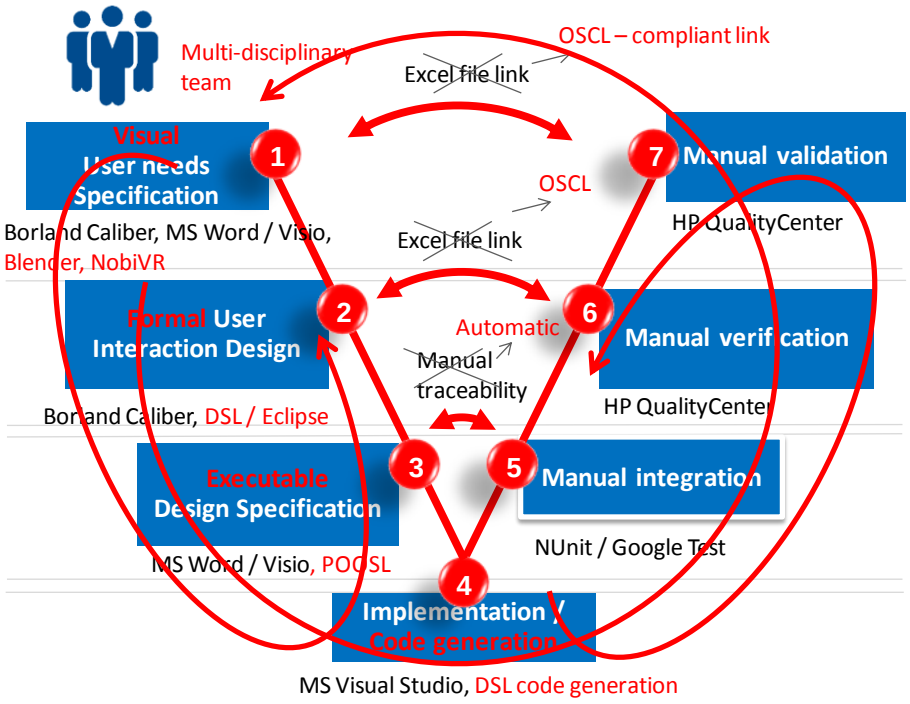


Figure 49 Proposed incremental development process and connections

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.

A.3 Scenario 1: Orientation of x-ray image on monitor

A.3.1 User Needs

The x-ray image on the monitor represents a two-dimensional 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.

A.3.2 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, cardioligst, surgeon). Because of the large set of variables, visualisation tooling is required to explore the real user needs and to find the optimal requirements.

A.4 Scenario 2: Setting x-ray beam projection with a joystick

User needs

Image-guided interventions and therapy demand for an eased workflow with regards to maneuvering table and stands. The integration with various components into the OR and Cathlab makes safe positioning of the X-ray system challenging. As an example, see the figure below where a Hybrid OR room is shown, full of equipment.

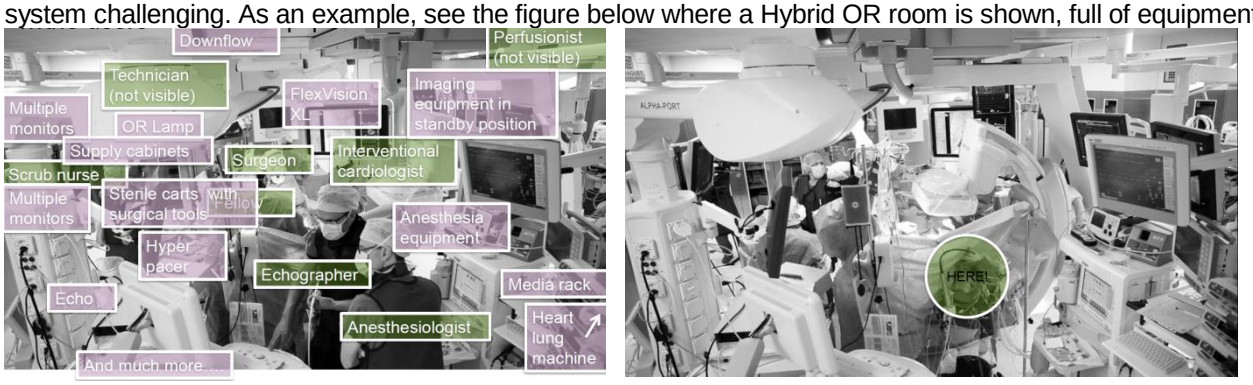


Figure 50 Hybrid OR with (left) all equipment and (right) the position of the patient in the room.

Ideally interventional X-ray camera's would be small and light, enabling easy control, not restricting in anyway the doctor in doing clinical procedures. Unfortunately this is not the case; in real life we have a heavy

camera (consisting of tube, collimator and flat detector, etc) which needs heavy and large mechanics. Moreover we need a large table to support and position our patient in any desired position.

Frontal Stand and Table

The picture below shows a frontal stand and table. The signs indicate whether the position increases (+) or decreases (-) for a movement in the direction of an arrow.

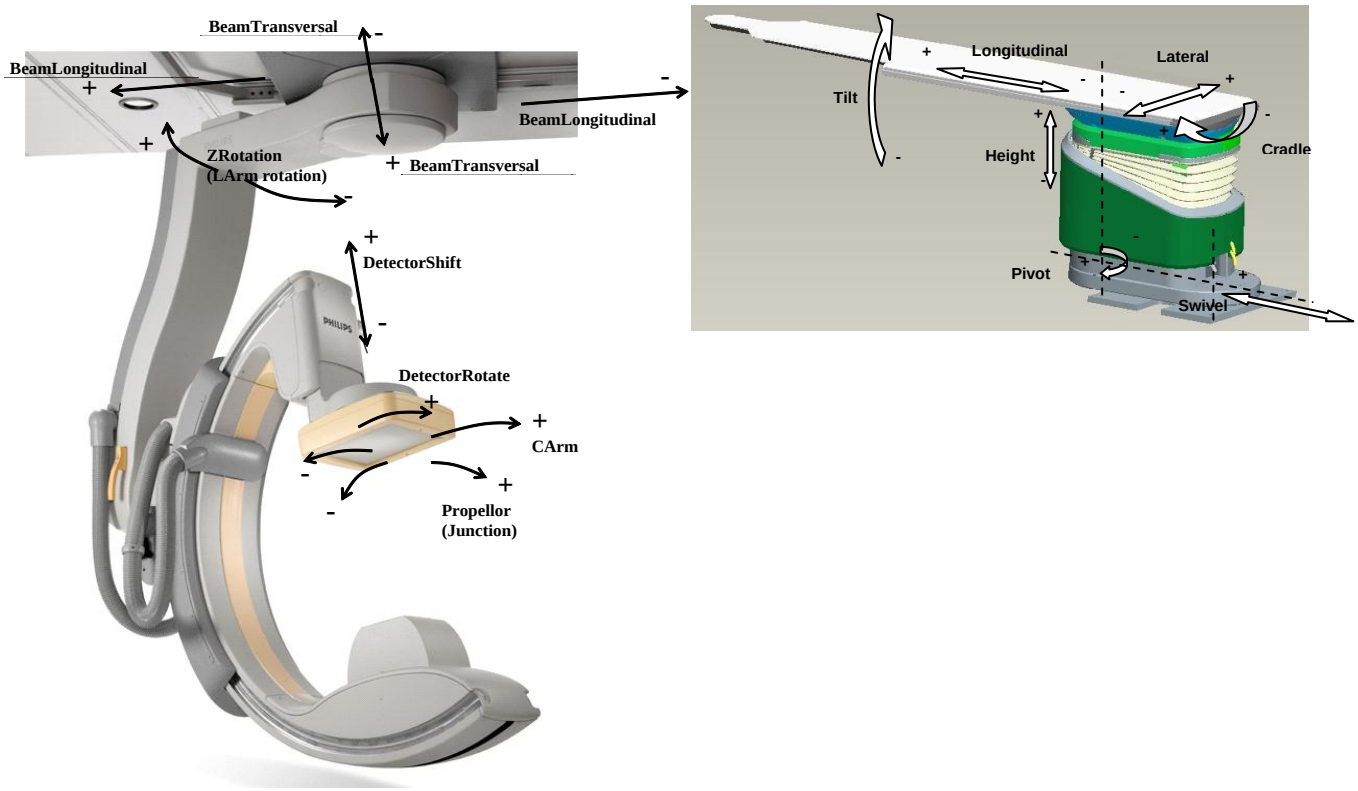


Figure 51: Frontal stand and table, illustrating the different axis in the system.

To improve the system on flexibility in patient setup, the introduction of multi axis movements is a dominant new feature for system behaviour.

Figure 52: patient accessibility is improved by multi axis movements.

The user interface shall control patient oriented movements and enable rotate, angulate with respect to the patient, see definition below.

3D definitions

The room, patient support, patient, X-ray and detector each have their own co-ordinate system. In this way, the relative positions of them can be defined.

The following picture shows these co-ordinate systems. Note that the patient coordinate system has been omitted in the picture.

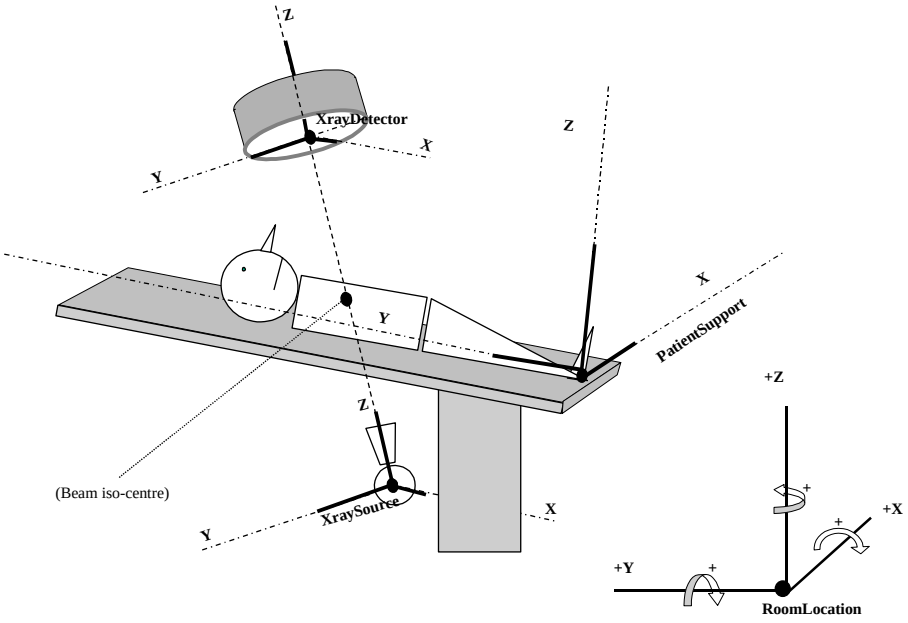
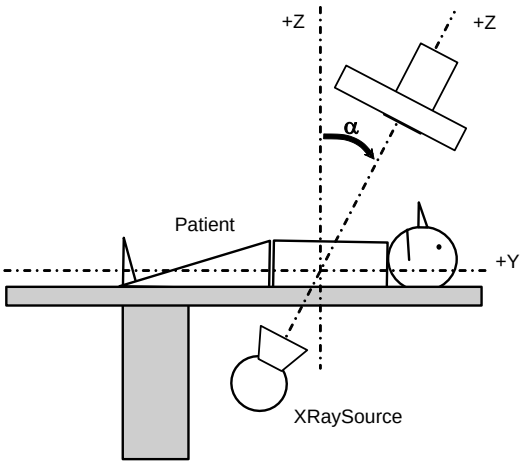
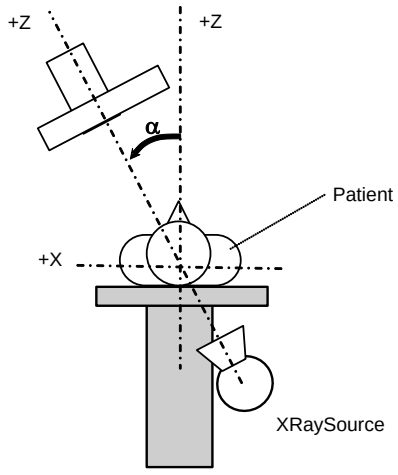


Figure 53: Coordinate systems for Allura positioning system.



PatientToXRaySource AngulationAngle = α

= –30° α



α = PatientToXRaySource RotationAngle

α = +30°

From application point of view, the system behaviour should be independent of the actual positions of the individual movement axes as much as possible. That is, the angulation and rotation movements should be independent of the actual position of the LArm or TablePivot. As a result, most user requests activate multiple movement axes and as such, no one-to-one relation exists (anymore) between user interface buttons/joysticks and the basic movement axes. These so-called patient orientation movements are movements where detector and collimator are aligned with lines of constant rotation and angulation and globe is aligned with patient axis (see picture).

Case study description

Several challenges appear at the horizon when designing a system where there is no one-to-one relation between user interface buttons/joysticks and the basic movement axes:

(1) Limitations of patient oriented movements by hardware restrictions. Philips wants to gain insight in and demonstrate the possibilities and limitations of the system to applicants or stake holders.
(2) Multiple scenarios where detector or collimator might collide with the table stand. The physical movement ranges of all individual axes are not limited such that no collisions can occur. In order to prevent collisions path guarding software checks for impending collisions. This way a certain clearing distance is taken into account. From an end users point of view the clearing distance should be as small as possible in order to get an optimal view on a patient. From a design point of view it is simpler and cheaper to make the clearing distance large. Philips wants to investigate the way the smallest possible clearing distance can be achieved between C-arc (with collimator and detector) and table stand in multiple scenarios given certain axes accuracies.

The above challenges result in the following technical case studies (WP4.1):

(1) Calculation of patient oriented movement limits caused by limitation of movement ranges of mechanics
(2) Calculation of patient oriented movement limits caused by stand-table collisions
(3) Interoperability of mathematical model with 3D visualization of stand and table
(4) Accuracy decomposition of stand- table collisions
(5) Calculation of detector oriented movement limits
(6) Use cases with table pivot not orthogonal
(7) Simulation of degraded functionality past patient oriented movement limits.

The case studies apply to a multitude of stand and table combinations, given the list below.

Multiple configurations of the stand:

- Different ceiling suspensions: Y, XY
- Different detector formats: FD20, FD15, FD12
- With and without spacer

Multiple configurations of the table

- AD7, Tilt, Cradle, Pivot
- Different tabletops: cardio, neuro
- External table / Maquet

A.5 Scenario 3: Movement direction of bolus chase

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.

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?

Appendix B Engineering methods

Engineering Method UC401_Formalize_UID

Engineering Method: UC_FormalizeUID

Purpose: The objective of this engineering method is to formalize the content of a User Interaction Design described in Natural Language (NL), and capture its content in an information model. A Domain Specific Language (DSL) is used as intermediate. The general concept here is a text-to-model transformation. The focus is on user-perceived behavior; thus suitable for visual verification and validation by domain experts. Aim is to bridge the gap between system design and detailed design/implementation.

Comments:

Pre-Condition	Engineering Activity as Steps	Post-Condition
- User Interaction Design description in Natural Language text	1. Explore the User Interaction Design and determine the central theme for which a model is made. Determine the model objective, scope, context, and level of abstraction. 2a. Isolate the general concepts. 2b. Capture the concepts and their interrelationships in a model. 2c. Verify and rework the model concepts based on feedback from domain experts. 3a. Define a Domain Specific Language (DSL) supporting the modelled elements. 3b. Reformulate the User Interaction Design in the DSL syntax. 3c. Resolve contradictions and omissions; experiment with it to get it complete, concrete, and concise. 4a. Determine the input requirements for building an interactive environment for simulating the model behavior. 4b. Determine the outputs required for visualizing the model behavior. 4c. Implement the logic for executing the interactive model simulation; connect the inputs, model, and outputs 4d. Create a simulation script for reviewing purposes that highlights the User Interaction Design behavior. 4e. Update the requirements engineering environment and have the simulation script and environment linked. 5. Verify and validate the model content with domain experts, using the interactive simulation environment.	- User Interaction Design described in DSL text - Simulation environment for model validation

Notes:

Notes:

Artefacts provided as input of the activity		Artefacts produced during of the activity		Artefacts which are the result of the activity	
Name	User Interaction Design	Name	Domain Specific Language	Name	User Interaction Design
Generic Type: (Tool or language independent type)	Formal Requirement	Generic Type: (Tool or language independent type)	Information model	Generic Type: (Tool or language independent type)	Information model
Shared Properties: (Information to be shared in interaction between steps)	- Requirement headline - Requirement reference ID - Requirement description - Requirement category tags - Requirement author(s) - Lifecycle status information	Shared Properties: (Information to be shared in interaction between steps)	- Model headline - Model description - Model category tags - Model author(s) - Model interfaces - Lifecycle status info.	Shared Properties: (Information to be shared in interaction between steps)	- Model headline - Model description - Model category tags - Model author(s) - Model interfaces - Lifecycle status info.
Description: 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.		Description: a set of grammer rules that make up a specific language.		Description: the model provides a simplified abstract representation that mimics complex behavior in the physical world. It is a logical arrangement of more elementary operations and variables. The language used can be as diverse as mathematics, statistics, logics, or UML. The topic modelled can be equally diverse and includes physics, economics, weather conditions, and others.	

Engineering Method UC401_Verify _Software_Architecture_Design

Engineering Method: UC_VerifySWArchitectureDesignUID			
Purpose: Pre-verification of software architecture and design on interface compliance for new or changed user interaction scenarios, without the need to early implement the feature, focusing on early-fault detection with less software effort, reducing thoughput time in projects for high-risk changes.			
Comments:			
	Pre-Condition	Engineering Activity as Steps	Post-Condition
	Input: High-level software architecture and interfaces described in (UML) drawing tool, User Interaction scenarios to verify are available	Step 1: High-level software architecture modeled in rapid-prototying environment (components / interfaces) Step 2: UI input devices connnection to rapid-prototying environment (joysticks, buttons etc) Step 3: Mechatronics of X-ray system (patient table, X-ray stand) visualized in 3D viewer Step 4: Motion output deviced connection to 3D viewer (movements, UI messages) Step 5: Iteratively perform: Step 5a: User intraction scenarios simulated in rapid-prototying environment (black-box approach) Step 5b: Adapt component dependencies based on modeling output (white-box approach) Step 5c: Determine impact on software archticture and interfaces. Step 6: Write test scenarios for regression Step 7: Write software change / PR to adapt modeling results in real software architecture.	Output: User interaction scenarios simulated in the rapid-prototying environment Next step(s): *High-level architecture design evaluated / verified against the new scenario and interfaces adapted where needed *Rapid prototying environment connected to real software architecture for mixed model-based/physicial execution
			Notes: Tools: Visio (UML drawing tool), POOSL (rapid-prototyping) Interoperability: See Visualize UID Stakeholders involved: Software architect and designers (early verification of design) Innovation: Using a rapid-prototyping environment for early verification of architecture design.

Engineering Method UC401_Verify_Requirements

Engineering Method: UC_VerifiyRequirement		
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 engineering and a proper test design, as missing or partially tested engineering 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.		
Comments:	Engineering Activity as Steps	Post-Condition
Pre-Condition - 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)	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). 7a. In RLCM the requirement with its linked test case and including the status is visible and can be reported on.	Verified Engineering Requirement + Authorized verification report containing the verified requirement(s) without manual push and pull interfaces and extra manual checks on data integrity and consistency.
	Next steps: Automatically set traceability and execute automated testing, up to creating a report of the test results.	Notes:

The table below maps the IOS services on the steps in the engineering method VerifyRequirements.

IOS Service	Brick Allocation	SP6 WP.x Allocation			Verify Requirement	Questions / Comments
<IOS-domain>.<actor>.<service>	<tool> / <brick-no>	<WP6.x>				\
	RLCM					|
OSLC_RM.consumer.read	IBM DOORS NG / B2.16	WP6.7				|
OSLC_RM.provider.create_link	TMAE					\
	HP-QC / B4.12	WP6.8				> **EXAMPLES**
OSLC_QM.provider.update	PLDM					/
	PTC Windchill	WP6.?				|
						|
						|
						/
none	RLCM				**1**	export document to PLDM? Should be a separate EM
OSLC_RM.consumer.read	PLDM				**1a**	propagate (push?) / from PLDM or RLCM?
OSLC_RM.provider.update	RLCM				**1a**	update status of baseline after review
OSLC_RM.consumer.update	PLDM				**1a**	update status of baseline after review
OSLC_RM.consumer. read	TMAE				**2**	
OSLC_RM.provider.basic_query	RLCM				**3**	open: template for report generation
OSLC_RM.consumer.basic_query	TMAE				**3**	open: template for report generation
none	TMAE				**4-1**	
OSLC_RM.provider.create_link	RLCM				**4-2**	alternative 1
OSLC_RM.consumer.create_link	TMAE				**4-2**	alternative 1
OSLC_RM.provider.resource_picker	RLCM				**4-2**	alternative 2
OSLC_RM.consumer.resource_picker	TMAE				**4-2**	alternative 2
OSLC_QM.provider.update	TMAE				**4a**	
OSLC_QM.consumer.update	RLCM				**4a**	called in RLCM when Req. is modified
OSLC_QM.provider.basic_query	TMAE				**5, 7a**	export document to PLDM?
OSLC_RM.provider.basic_query	RLCM				**5**	export document to PLDM?
OSLC_QM.consumer.basic_query	document generation				**5**	
OSLC_RM.consumer.basic_query	document generation				**5**	
not_yet_defined.consumer.store_in_archive	document generation				**5**	
not_yet_defined.provider.store_in_archive	PLDM				**5**	
none	TMAE				**5**	for TMAE internal queries (e.g. TD/TC overview)
OSLC_QM.consumer.update	PLDM				**6**	unclear what's meant by triggers
OSLC_QM.provider.update	TMAE				**6a**	
OSLC_QM.consumer.update	PLDM				**6a**	action initiated form PLDM?
OSLC_QM.provider.read	TMAE				**7**	
OSLC_QM.consumer.read	RLCM				**7a**	
Tool Chain Specification for Deployment and Tailoring						
Additional notes on the EMs						
Domain-agnostic?					**generic**	
Complexity of the EM					**simple**	
Comments						
Existing Engineering Standards to be potentially used						
~~Mapping onto existing IOS OSLC Specs~~					~~**OSCL RM/ OSLC QM/ OSCL CM**~~	
Potential IOS Extensions					**store retrieve to archive,**	1
Potential IOS Extensions					**common report template support**	1
Potential IOS Extensions					**automatic flagging of changes**	4a

The EngineeringMethod VerifyRequirements have been extended after discussions with the IOS specialist and brickowners.

Engineering Method: UC_VerifiyRequirement			
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			
Comments:			
Pre-Condition	Engineering Activity as Steps		Post-Condition
- Applications that can share data in its context - Requirements are available - Tests are available	0. Being part of a regulated business it is needed to archive design evidence (known as Design History Files or short DHF) and maintain the DHF 15 years after the last product is delivered, for efficiency purposes (cost, IT etc.) harmonized archives are used of which PLDM (Product Lifecycle Data Management) is one.		Verified Engineering Requirement + Authorized verification report containing the verified requirement(s) without manual push and pull interfaces and extra manual checks on data integrity and consistency.
	1a. Create requirements in an application for Requirements Lifecycle Management (RLCM) and make a requirements document / content / baseline available in the PLDM. Tools that have not been designated as an archive need to be able to export the content to the designated archive. In the PLDM system the review and approvals on content are done.	Requirements Engineer	
	1b. Other applications should be able to retrieve the status of the requirements from the PLDM system. Status is "Preliminary" or alike, "in Review" or alike or "Authorized" or alike.	Configuration Manager	
	1c. The requirements in the report / baseline which are reviewed and approved in the PLDM system should have an updated status.	Software application	
	2a. The RLCM environment is able to generate a requirements report needed for evidence logging. This report is created via baselines. The baselines contains a set of requirements selected by the System Designer.	Requirements Engineer System Designer	
	2b. The baseline should be available for other applications to use (TMAE).	Software application	
	3. 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).	Test Designer	
	4a. A Test Design (defining test approach, which environments to use and what configurations are covered) is created for a cluster of related requirements;	Test Designer	
	4b. Requirements are translated into Test Cases while their relationship is set to both requirements and Test Designs for requirements-to-test traceability purposes (aka. Test Tracability Matrix).	Test Designer	
	4c. 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.	Requirements Engineer System Designer Test Designer	
	4c1. Un update in a requirement is logged in the history of the requirement (Check-out / check-in), no matter how small the change is, an alert is provided to linked data, in this case a Test Design and or Test case (1:n).	Software application	
	4c2. Preferably different flags are available, atleast one for a pending change and one for a new approved version.	Software application	
	4c3. In the TMAE a flag or alert is visible in the overview of all test cases and in the details of the test case. With 1 press of a button the changes in the requirment should be available / visible to the user.	Test Designer	
	4c4. When the user in the TMAE indicates the changes in the Requirement have been analyzed and implemented the flag should be cleared / archived.	Test Designer	
	5a. 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	Test Designer	
	5b. The reports are directly available in the PLDM system for the review, approval and archiving process.	Software application	
	6a. As soon as approvals in the PLDM application are given the updated status should be available in the linked applications.	Configuration Manager	
	6b. In the Test management & execution software the test designs and cases receive an updated status.	Software application	
	6c. The updated status should be accompanied by a content check / version check (check if the content of the test case is changed during the review and authorization process).	Software application	
	7a. Test Execution can be started and will result in a status update in traceability and other applications.	Test Engineer	
	7b. In RLCM the requirement with its linked test case and including the traceability status is visible and can be reported on.	System Designer Test Engineer	
	Next steps: Automatically set traceability and execute automated testing, up to creating a report of the test results.		Notes:

Artefacts provided as input of the activity		Artefacts produced during of the activity		Artefacts which are the result of the activity	
Name		Name	Document	Name	Traceability matrix
Generic Type: (Tool or language independend type)		Generic Type: (Tool or language independend type)	Technical documentation	Generic Type: (Tool or language independend type)	Technical documentation
Shared Properties: (Information to be shared in interaction between steps)		Shared Properties: (Information to be shared in interaction between steps)	- Document title - Document type - Document reference ID - Document author(s) - Lifecycle status information	Shared Properties: (Information to be shared in interaction between steps)	- Req. headline - Req. outcome - Test headline - Test outcome - Req. outcome status info
Description:		Description: individual document or report, consisting out of one or more files in formats supported by the engineering environment. Information is subject		Description: compiled overview where, for each requirement	
Name		Name	Requirement	Name	
Generic Type: (Tool or language independend type)		Generic Type: (Tool or language independend type)	Formal Requirement	Generic Type: (Tool or language independend type)	
Shared Properties: (Information to be shared in interaction between steps)		Shared Properties: (Information to be shared in interaction between steps)	- Requirement headline - Requirement reference ID - Requirement description - Requirement category tags - Requirement author(s) - Lifecycle status information	Shared Properties: (Information to be shared in interaction between steps)	

Appendix C Tool Chain Description

This paragraph provides a short introduction on the individual engineering tools (aka. Brick) mentioned in the Systems Engineering Environment at M0 and M12, and the tool interoperability capabilities offered.

Electric Cloud – ElectricCommander

ElectricCommander automates and accelerates software delivery using a engine that unites production processes and their supporting IT resources. These processes typically include building, testing, releasing, and deploying software. It provides complete IT resource management through support for physical, virtual, and public/private cloud infrastructure. The platform provides hundreds of integrations to industry tools like compilers, test systems, code coverage tools, infrastructure platforms, etc.

Artifacts:
- In: Build Request
- Out: Build Result
- In: Test Request
- Out: Test Result

OSRF – GAZEBO (B4.10)

The Gazebo tool combination provides modeling of performance of key system parameters and adopting a multi-physics approach. Interactions between key parameters can be explored before a detailed design or hardware is made, allowing for cost-effective and flexible system definition.

Google – Google Test

Google's framework for writing C++ tests on a variety of platforms (Linux, Mac OS X, Windows, Cygwin, Windows CE, and Symbian). It is based on the xUnit architecture and supports automatic test discovery, a rich set of assertions, user-defined assertions, death tests, fatal and non-fatal failures, value- and type-parameterized tests, various options for running the tests, and XML test report generation.

Artifacts:
- In: Test
- Out: Test Result

Mathworks – Matlab/Simulink (B3.46)

Simulink is a popular dynamic systems modeler with a broad scope of features, rich ecosystem and wide use. Simulink is found in many domains and is the source for complex model-based tool-chains in software development for embedded systems. It is the de-facto industry standard in simulation model development.

Artifacts:
- In: Models (on data flow in dynamic systems)
- In: Test (signal data and the simulation script)
- Out: Source Code
- Out: Test Results

NUnit.org – NUnit

NUnit is a unit-testing framework for all .Net languages. It is part of the xUnit based unit testing tool for Microsoft .NET. NUnit has two different ways to run your tests. The console runner, nunit-console.exe, is the fastest to launch, but is not interactive. The gui runner, nunit.exe, is a Windows Forms application that allows you to work selectively with your tests and provides graphical feedback

Artifacts:
- In: Source Code
- In: Test

- Out: Test Result

SourceWorks – OROCOS (B4.11)

Orocos is a tool chain, dedicated to Open RObot COntrol Software development for Model-Driven Engineering. It is fully component-based and multi-vendor. This tool helps creating real-time robotics applications using modular, run-time configurable software components.

Artifacts:
- In: Model (of kinematics and dynamics)
- In: Source Code
- Out: Executable Code (e.g. packed in an install shield)

HP – Quality Center (B4.12)

HP Quality Center will be used by this brick in the context of requirement management, test case management and traceability, test case execution, creating of report information, SW release management. HP Quality Center should be integrated to the interoperability standard (IOS) in a corresponding manner.

Artifacts:
- a

IBM – Rational ClearCase

ClearCase is a software configuration management solution that provides version control, workspace management, parallel development support, and build auditing. You can integrate Rational ClearCase with other IBM solutions, including IBM Rational Team Concert, IBM Rational ClearQuest, IBM Rational Asset Manager, and IBM Rational Application Developer for WebSphere Software. Rational ClearCase scales to any size team from small workgroup to large, geographically distributed teams.

Artifacts:
- In: Requirements
- In/Out: Tests
- In/Out: Test Results
- In/Out: Test Environment Details

IBM – Rational ClearQuest (B3.87)

A Change Request system, which controls the flow of information wrt to any external or internal change requests after a freeze of requirements. This is essentially also a partial requirements database – however the information incoming from here needs to be transferred into the ReqPro database with data integrity and all attributes intact. Currently there is no common interface so IOS should be investigated.

Artifacts:
- In/Out: Change Request

IBM – Rational DOORS Next Generation (B2.16)

IBM Rational DOORS is a widely adopted product, system and software requirements management tool to enable requirements communication, collaboration, and verification. It is optimized for the needs of complex and embedded systems development, and is a candidate to be considered for prototyping of an IOS interface.

Artifacts:
- In/Out: Requirements

IBM – Rational Quality Manager

Quality Manager is a web-based centralized test management environment that provides a collaborative and customizable solution for test planning, workflow control, tracking and metrics reporting. It acts as collaborative hub for business-driven software and systems quality across virtually any platform and type of testing. This software helps teams share information seamlessly, use automation to accelerate project schedules and report on metrics for informed release decisions.

IBM – Rational Rhapsody (B2.10)

IBM Rational Rhapsody is a widely adopted family of tools targeting towards visual, model-driven development for systems and software applications. It provides collaborative design and development for systems engineers and software developers creating real-time or embedded systems and software, with support for dependable systems including safety, security and reliability.

Artifacts:
- In/Out: Models (on structure and behavior)
- In/Out: Source Code

IBM – Rational Team Concert (B2.19)

IBM Rational Team Concert is a widely adopted systems and software lifecycle management solution that enables real-time, contextual collaboration for distributed teams. It includes agile, formal and hybrid planning and reporting, with support for the automation of complex and embedded systems development and powerful collaborative change management capabilities. RTC is a candidate to be considered for prototyping of an IOS interface.

Artifacts:
- In/Out: Source Code
- In/Out: Executable Code
- In/Out: Change Request

Microsoft – Visual Studio

Visual Studio is a comprehensive collection of tools and services for developing applications that target the desktop, the web, devices, and the cloud. It provides an integrated development environment (IDE) and collaboration environment that welcomes connection with other development tools, such as Eclipse and Xcode. It leverages Visual Studio's state-of-the-art development environment for .NET languages, HTML/JavaScript, and C++ for teams working across multiple platforms.

Artifacts:
- Model of user interface (when applied)
- Source code (both input and output)
- Executable code
- Project file

Philips – Xposer

Xposer is a proprietary tool developed by Philips for visualizing and simulating positioning system movements. It utilizes Ogre for 3D visualization and rendering, and QT for the modelling a user interface. The physical configuration of the positioning system is described using XML. OGRE (Object-Oriented Graphics Rendering Engine) is a scene-oriented, flexible 3D engine written in C++ designed to make it easier and more intuitive for developers to produce applications utilising hardware-accelerated 3D graphics. The class library abstracts all the details of using the underlying system libraries like Direct3D and OpenGL and provides an interface based on world objects and other intuitive classes. Qt provides different approaches for developers and designers to create application user interfaces and gives the freedom to select the best workflow and UI approach for the development purposes.

Artifacts:
- Model of physical positioning system, with its axes and degrees of freedom.

Eclipse.org – Xtend (B4.04)

Xtend is a statically-typed programming language which translates to comprehensible Java source code. Syntactically and semantically Xtend has its roots in the Java programming language but improves on many aspects. Xtend is much more concise, readable and expressive than Java. Xtend's small library is just a thin layer that provides useful utilities and extensions on top of the Java Development Kit (JDK). The compiled output is readable and pretty-printed, and tends to run as fast as the equivalent handwritten Java code.

Artifacts:
- Source code (both input and output)
- Project file

Eclipse.org – Xtext (B4.04)

Xtext is a framework for development of programming languages and domain specific languages. It covers all aspects of a complete language infrastructure, from parsers, over linker, compiler or interpreter to Eclipse IDE integration. It provides a set of domain-specific languages and APIs to describe the different aspects of your programming language. Based on that information it gives a full implementation of that language running on the JVM. The compiler components of the language are independent of Eclipse or OSGi and can be used in any Java environment. They include such things as the parser, the type-safe abstract syntax tree (AST), the serializer and code formatter, the scoping framework and the linking, compiler checks and static analysis aka validation and last but not least a code generator or interpreter. These runtime components integrate with and are based on the Eclipse Modeling Framework (EMF), which effectively allows the use of Xtext together with other EMF frameworks like for instance the Graphical Modeling Project GMF.

Artifacts:
- In: Model (of DSL language)
- In: Source Code
- Out: Executable Code (of DSL parser)

NobiVR

NobiVR is a virtual reality (VR) layer for other applications to use. Its goal is to provide an abstraction over different types of VR technology and handle the technical details that are required to provide application users with an immersive virtual reality experience.
The NobiVR layer accommodates 3D motion tracking input for natural user interaction, and multiple types of 3D visualisation configurations (ranging from passive and active 3D screens to head mounted displays).

Ogre 3D

Ogre (Object-oriented Graphics Rendering Engine) is an open-source graphics rendering engine that is written and maintained by a small core team, and contributed to by its (ever growing) community. Ogre is a real time rendering engine, which implies that each image is rendered in 1/30th to 1/100th of a second. Ogre runs on a wide variety of hardware capable of 3D graphics.