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

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

Specification, Development and Assessment for Heterogeneous Simulation

D606.011

DOCUMENT INFORMATION

Project	CRYSTAL
Grant Agreement No.	ARTEMIS-2012-1-332830
Deliverable Title	Specification, Development and Assessment for Heterogeneous Simulation
Deliverable No.	D606.011
Dissemination Level	CO
Nature	R
Document Version	V1.0
Date	2014-01-31
Contact	Thomas Kuhn
Organization	FhG IESE
Phone	+49 631 6800 2177
E-Mail	thomas.kuhn@iese.fraunhofer.de

AUTHORS TABLE

Name	Company	E-Mail
Thomas Kuhn	Fraunhofer IESE	thomas.kuhn@iese.fraunhofer.de
Sören Schneickert	Fraunhofer IESE	soeren.schneickert@iese.fraunhofer.de
Bardo Bakker	TNO	bardo.bakker@tno.nl
Jean-Luc Johnson	EADS	Jean-Luc.Johnson@eads.com

CHANGE HISTORY

Version	Date	Reason for Change	Pages Affected
0.1	20.09.2013	Initial Outline	All
0.2	03.12.2013	Added FhG IESE context (Kuhn)	All
0.3	19.12.2013	Added content to chapter 4.3 (Schneickert)	All
0.4	27.12.2013	Added Barco Use-Case	All
0.9	27.12.2013	Added paretner context	All
0.91	15.01.2014	Internal Review (Johnson, Kuhn, Schneickert)	All

CONTENT

D6.6.1-1 I

1 INTRODUCTION 7

1.1 ROLE OF DELIVERABLE 7
1.2 RELATIONSHIP TO OTHER CRYSTAL DOCUMENTS 7
1.3 STRUCTURE OF THIS DOCUMENT 7

2 REQUIREMENTS FROM RELEVANT USE-CASES 8

2.1 USE CASE DESCRIPTION 8
2.1.1 Use Case 2.08 8
2.1.2 Use Case 4.05: A software centric scalable safety critical medical display platform 9
2.1.3 Use Case 4.06: An intelligent infusion controller for blood pressure regulation in operating room 11
2.2 WHAT ARE EXPECTATIONS OF USE-CASES TO WP 6.6 BRICKS 13
2.2.1 Use Case 2.08 13
2.2.2 Use Case 4.05: A software centric scalable safety critical medical display platform 14
2.2.3 Use Case 4.06: Infusion controller for blood pressure 15

3 RELATED WORK 17

4 SPECIFICATION OF DEVELOPED SOLUTIONS 18

4.1 APPROACH FOR INTEGRATION OF HETEROGENEOUS MODELS (B2.47) 18
4.1.1 Brick description 18
4.1.2 Addressed requirements from Use-Case 18
4.1.3 Proposed solution 19
4.1.4 Relationships to dependent bricks 19
4.2 SIMULATIVE EVALUATION OF HETEROGENEOUS MODELS (B2.48) 19
4.2.1 Brick description 19
4.2.2 Addressed requirements from Use-Case 20
4.2.3 Proposed solution 20
4.2.4 Relationships to dependent bricks 20
4.3 MODEL-BASED DESIGN VERIFICATION METHOD (B2.49) 21
4.3.1 Brick description 21
4.3.2 Addressed requirements from Use-Case 21
4.3.3 Proposed solution 21
4.3.4 Relationships to dependent bricks 21
4.4 MODELICAML TOOL FOR DESIGN VERIFICATION BASED ON MODELS (B2.40) 21
4.4.1 Brick description 22
4.4.2 Addressed requirements from Use-Case 22
4.4.3 Proposed solution 22
4.4.4 Relationships to dependent bricks 22
4.5 SIMULATION WITH HARDWARE IN THE LOOP CAPABILITIES (B4.06) 22
4.5.1 Brick description 22
4.5.2 Proposed solution 26
4.5.3 Relationships to dependent bricks 27
4.6 REAL-TIME HIL FOR CRITICAL FEATURES (B4.17) 27
4.6.1 Brick description 27
4.6.2 Addressed requirements from Use-Case 27
4.6.3 Proposed solution 28
4.7 PERFORMANCE SIMULATION (B4.09) 30
4.7.1 Addressed requirements from Use-Case 30
4.7.2 Proposed solution 31

4.7.3 Relationships to dependent bricks *31*

5 CONCLUSION **32**

5.1 CONCLUSION 32
5.2 TERMS, ABBREVIATIONS AND DEFINITIONS 32

6 REFERENCES **33**

Content of Figures

Figure 2-1: Instantiating a hybrid simulation scenario 8
Figure 2-2: Typical Medical Display Image Pipeline 9
Figure 2-3: Current V model process 10
Figure 2-4: Blood Pressure Schematic 11
Figure 2-5: Intensive care unit 12
Figure 2-6: Tools to be used in the V-Model (first analysis) 13
Figure 4-1: Example (automotive) scenario 18
Figure 4-2: Heterogeneous scenario 20
Figure 4-3: Position of various engineering methods in the development cycle, as taken from CRYSTAL_D_406_010_v1_0.doc 23
Figure 4-4: graphical representation of the interconnections between the engineering methods, connected to the engineering methods as given in Figure 4-3 27
Figure 4-5: Interactions of real-time HiL 29
Figure 4-6: Real-time HiL design 30

Content of Tables

Table 2-1 14
Table 2-2 16
Table 4-1: Requirements for HiL simulation and for interoperability between HiL simulation and other engineering methods in UC406 24
Table 4-2 28
Table 4-3 30
Table 4-4 31
Table 5-1: Terms, Abbreviations and Definitions 32

1 Introduction

1.1 Role of deliverable

This deliverable contains the specification, development and assessment of all bricks of its corresponding work package. Each brick shall be represented in seperate chapters of this deliverable. The document will be released three times throughout the project duration whereof the first will contain the specification part.

Therefore the main purpose of this deliverable is the specification of requirements from bricks and use-cases, and how they match to contributions. Hence this deliverable documents what was made and why. The evaluations and evaluation results of bricks that have been developed in WP 6.6 will be documented in subsequent revisions of this deliverable as well. Scientific and technical results are documented in deliverable D.6.6.2.

1.2 Relationship to other CRYSTAL Documents

This deliverable has a strong relationship with deliverable D6.6.2, which describes the technical and scientific details of the realization of each contribution.

1.3 Structure of this document

The remainder document is structured as following: Section 2 documents relevant use-cases for this deliverable and collects relevant requirements. Section 3 surveys related work. Section 4 gives an overview on proposed realization approaches that WP 6.6 activities will focus on. Section 5 draws conclusions.

2 Requirements from relevant USE-Cases

2.1 USE Case description

2.1.1 Use Case 2.08

One of the main priorities of user Story 2.08 is the simulation of heterogeneous models. Today, model driven development approaches yield numerous self-contained and isolated models that focus on specialized aspects. These models are often created in different tools; therefore, the integration of these models is challenging. An early evaluation of system level properties requires the linking of different models into a heterogeneous simulation scenario. In user story 2.08, this will be demonstrated in context of a de-icing system. The simulation of this system shall evaluate the performance of three different de-icing techniques: electrical, pneumatical, and chemical.

Performance evaluation requires the linking of the functional models of each de-icing technique with environment models. Functional models are realized in Modelica, while environmental models are realized with Simulink, and controller models are provided by Rational Rhapsody. The goal of this user story is to enable a holistic simulation with an integrated prototype that is constructed by linking necessary simulation models. Ideally, models will be described by meta data attributes that document the purpose of a model, its interface, and necessary prerequisites. Developers are able to query this information and construct a holistic simulation scenario by instantiating these models. (cf. Figure 2-1)

This instantiation should be possible within the same organization, but ideally also across the borders of one organisation to integrate models from other departments, locations, or to integrate models from OEMs and suppliers into virtual prototypes.

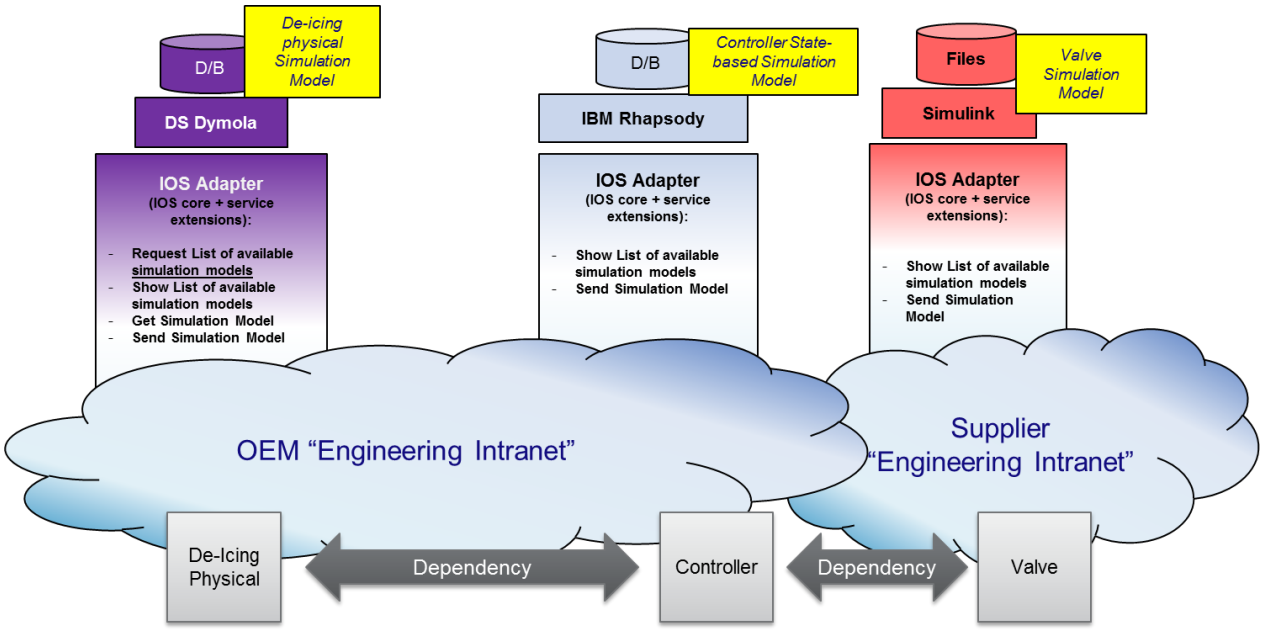

Figure 2-1: Instantiating a hybrid simulation scenario

2.1.2 Use Case 4.05: A software centric scalable safety critical medical display platform

There is ever increasing product variability and the need to speed up time-to-market and reduce development time. This requires Barco to change their medical display platform from a hardware centric, custom platform towards a flexible software centric, Commercial-off-the-Shelf (COTS) platform.

2.1.2.1 Current practice

Barco's current display platform is based on proprietary hardware using Field Programmable Gate Arrays (FPGA). This is cost-effective when there are only a limited number of products to be supported, but it requires a huge R&D effort to develop and maintain when there are an increasing number of product variants. Hardware development typically also requires longer development cycles.

In order to display a medical image correctly, an input signal (typically coming via a DVI input or Display Port input) needs several transformations before it can be shown to a radiologist. This sequence of transformations is the so-called image pipeline.

The output image needs to comply with several FDA regulations (e.g., DICOM GSDF, GrayScale Display Function). Figure 2-2 shows a diagram that gives a high-level overview of a medical display image pipeline.

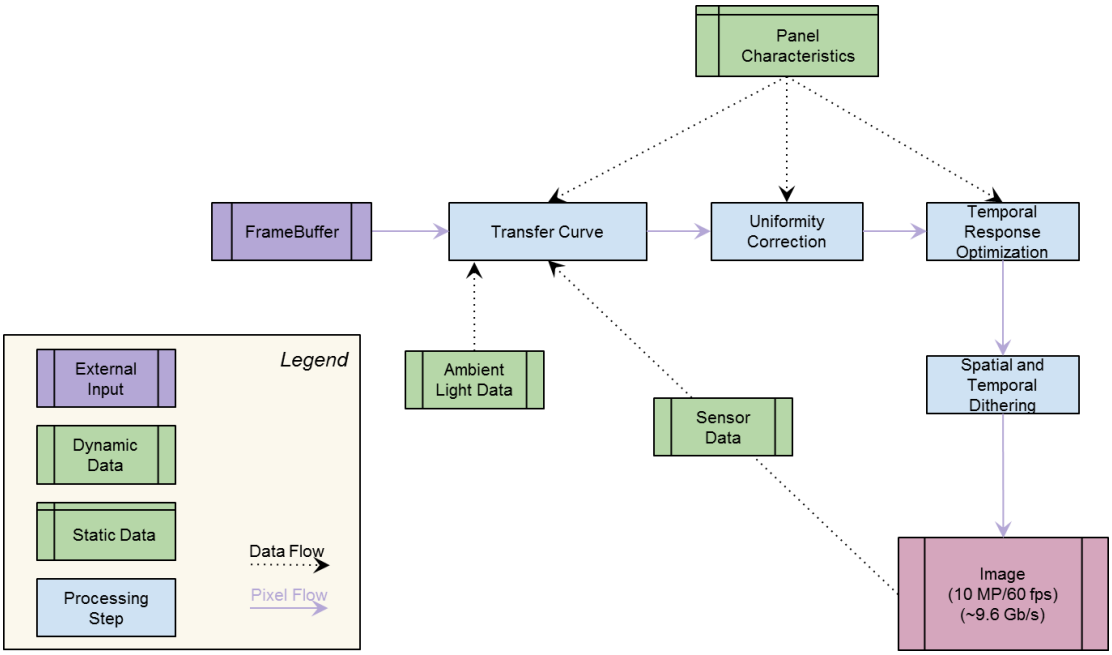

Figure 2-2: Typical Medical Display Image Pipeline

We would like to replace the proprietary FPGA-based hardware platform by a COTS platform while keeping the same level of safety and performance.

Currently, the image pipeline (as shown in Figure 2-2) is implemented using hardware components (i.e. FPGAs). With this use case, we want to achieve a software-only implementation of the processing steps marked in blue in Figure 2-2; the other steps remain dedicated hardware components. The processing steps from the figure are called Software Imaging Components. In a first step, the components are developed individually on COTS hardware.

By changing the design methodology from hardware-centric to software-centric we make it possible to create product variants by simply changing software configurations rather than writing new code or recompiling the code for new target hardware.

2.1.2.2 Current Software Process

The current software development process (excluding display development) is shown in Figure 2-3.

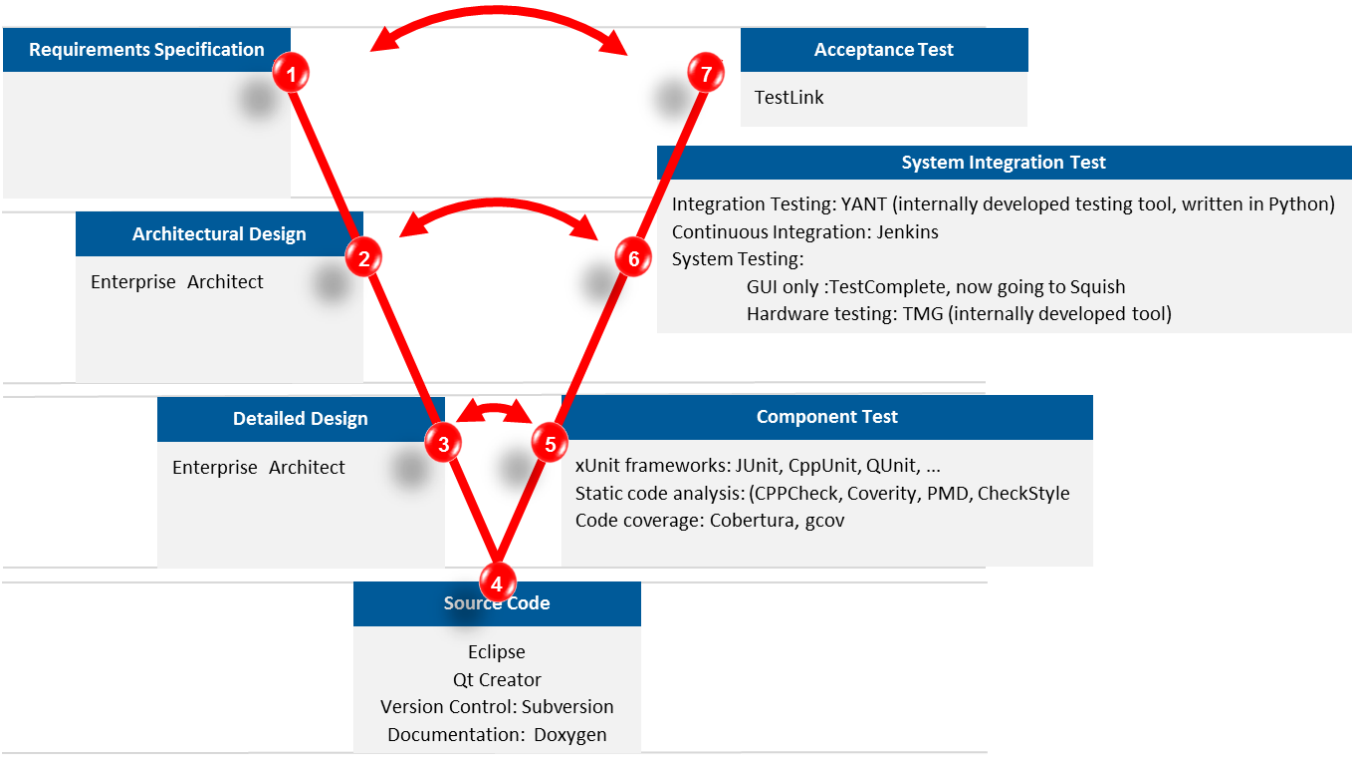

Figure 2-3: Current V model process

In the requirements specification step we gather the stakeholder requirements. No dedicated requirement tool is used for this step, this is currently done by using Excel and Word files. Based on the Excel and Word files, the system model and use cases are designed in the Architectural Design & Detailed Design steps, the output is an Enterprise Architect file. In the current process we are missing traceability to the requirements and we are not supporting the re-use of existing designs or components.

In the Source Code step we implement the functional code without having traces to the model or requirements. Concerning re-use, ad hoc, there is often source level reuse of software components. Developers will use code to test the coverage of the functional code during Component Test. In the current process we are missing coverage checks on requirements.

QA testers implement a System Integration Test based on use cases; these tests also include reused software components. Also in this step we have no link back to requirements.

In the final step QA testers implement Acceptance Tests based on requirements without having a solid trace to the original requirements.

If we want to integrate the development of a complete display, the current process needs to be extended/enhanced, resulting in a desired process.

2.1.2.3 The Desired Process

The major drawbacks of the current process are:

- The lack of traceability of requirements.
- No integration of the tools used during development.
- No reuse of models.
- No real reuse of components.
- Integration tests include testing reused software level components.
- Coverage check of unit tests, integration tests and acceptance tests towards requirements.
- No uniform, centralized documentation of the created artefacts.

However, due to the ever increasing regularization within the Healthcare market, documentation, traceability and auditability are becoming more and more important for product development. On the other hand, from an economic perspective, time-to-market requirements are becoming stricter and stricter.

Using model-based engineering, we hope to reduce the costly hardware-development cycles, so that eventual bottlenecks and/or problems can be detected much earlier and much faster.

Using tools that 'understand' each other, at least we hope to introduce requirements traceability throughout our development process, leading to consistent traceable documentation, as required by ISO 62304.

Getting approved for FDA, takes a lot of paper work and a lot of time. FDA-approval turnaround time can be reduced by reusing formerly approved components/designs into the development process.

2.1.3 Use Case 4.06: An intelligent infusion controller for blood pressure regulation in operating room

The goal of use Case 4.06 is to incorporate tools to support certain phases of the development of an intelligent infusion controller.

The product to be obtained in this process is a system that operates delivering vasoactive drugs with the ultimate goal of reducing patient´s hypertension, and controlling blood pressure measurements in a patient undergoing surgical intervention in OR (operating room) or in post cardiac surgery in ICU (intensive care unit).

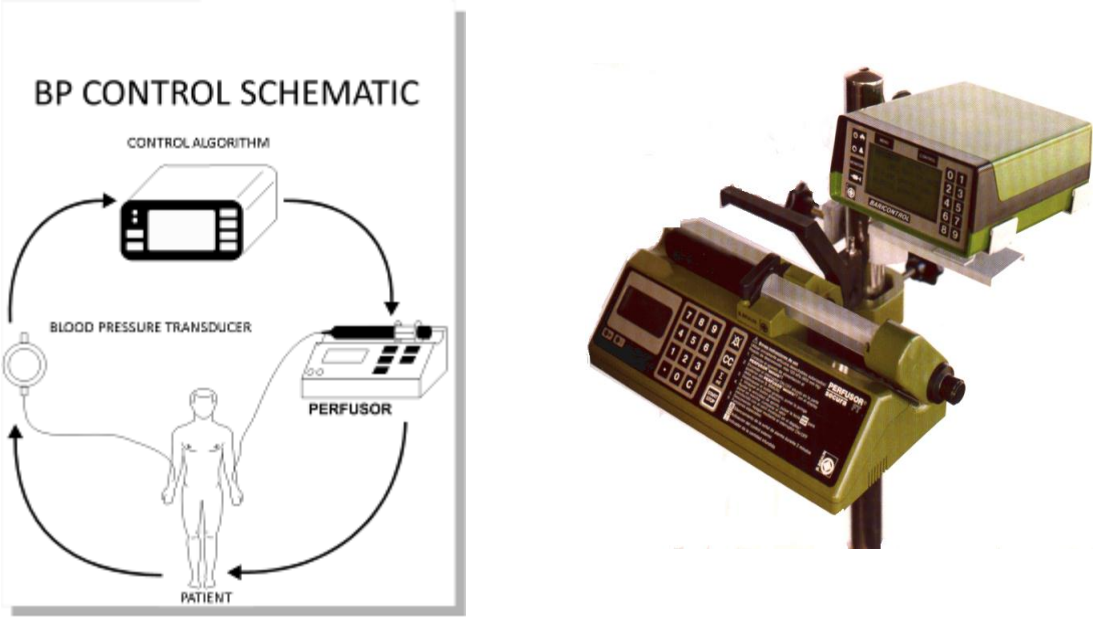

Figure 2-4: Blood Pressure Schematic

Need of the Product to be developed in the Use Case

As it has been said before the Use Case 4.06 process development is intended for an intelligent infusion controller. Hypertension occurs frequently in the immediately postoperative period after cardiac surgery, in spite of adequate analgesia and sedation. The usual management of this hypertension is by infusion of quick acting and ultra-short response vasodilators.

At present there are no or little technological alternatives practiced. The only noticeable alternative is the employment of medical assistants concentrated on the delivery of drug, while doing a lot of other things simultaneously. The probability of human error is quite high and the project would contribute to improve working quality of the clinical staff and the patients' safety.

Required Components

The system design includes and integrates from component level:

- A Multiparameter Vital Signs Monitor
- An Infusion Pump tree, as fully integrated accessory.
- Safe communication between the above mentioned components.
- A control algorithm
- Connectivity means to an I. System

The device system performance combines:

- Diagnosis and therapeutic capabilities,
- Means for enhancing patient care
- Means for releasing the nursing work, so that the clinical staff can have more time to focus on other demanding areas.

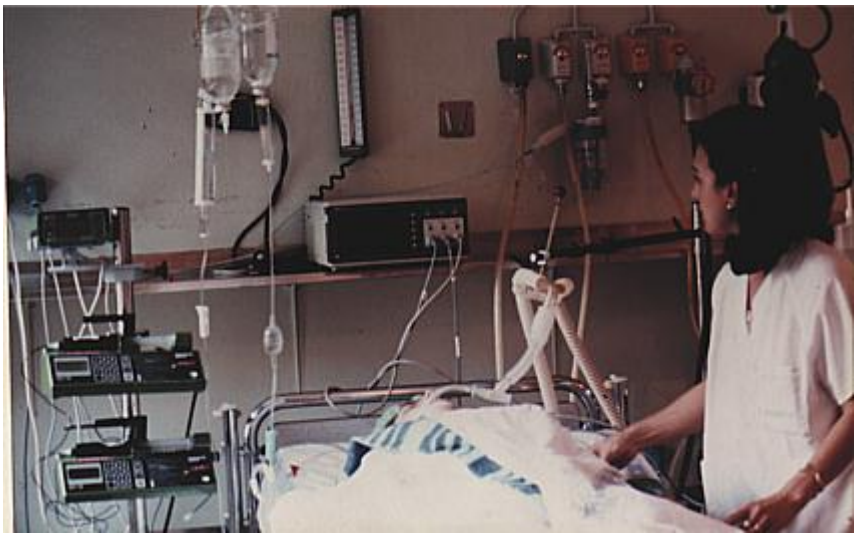

Figure 2-5: Intensive care unit

Use case IOS Needs

IOS needs in the US4.06 tool chain are mainly related to facilitate the requirements traceability along the whole V process, and more specially with validation and simulation activities for shortening development times and assisting in the certification phase of the product.

In Figure 2-6 it can be seen the identified tools in a first analysis to be used in the V model of the process development for the intelligent infusion controller.

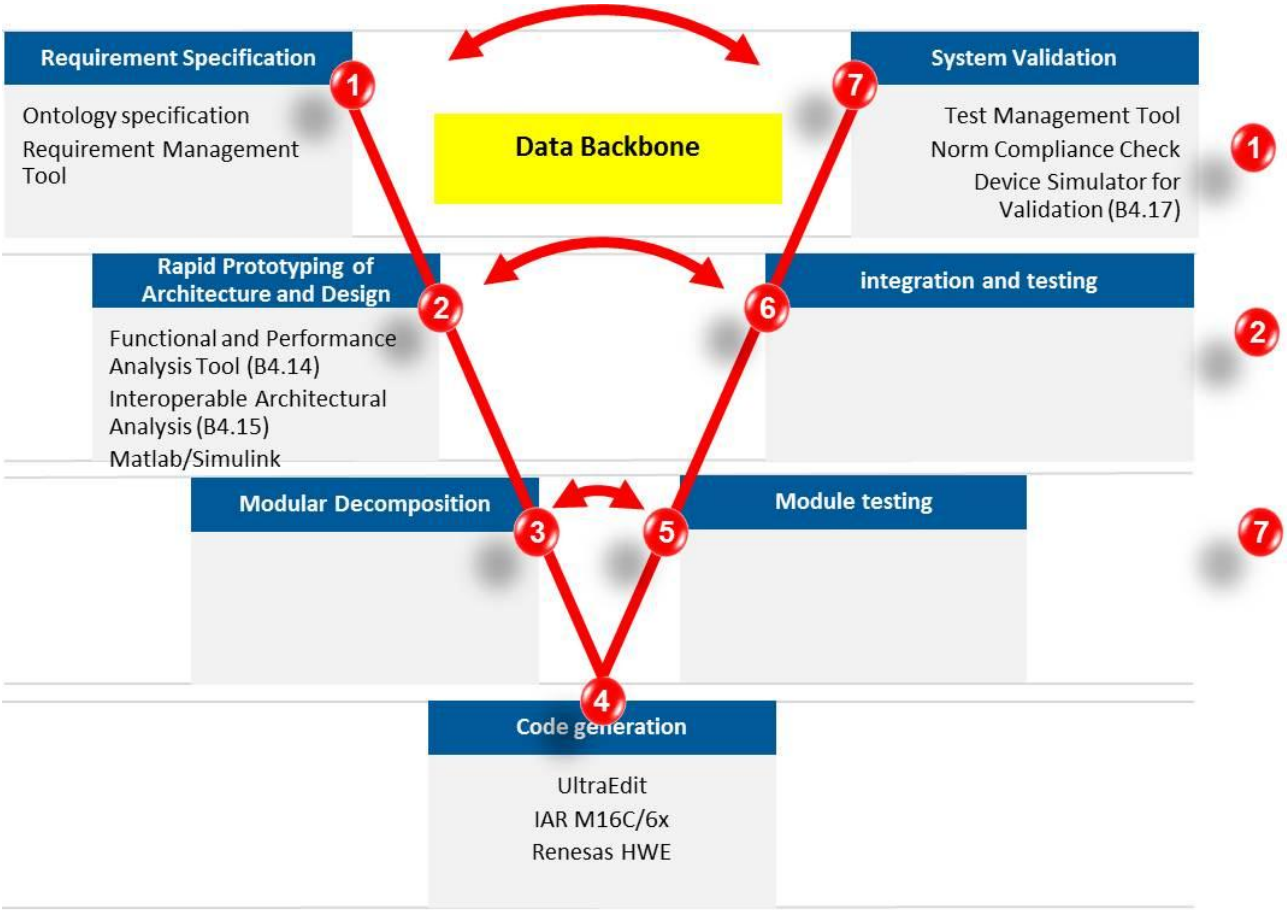

Figure 2-6: Tools to be used in the V-Model (first analysis)

For more information on this US, please refer to D4.6.1; This is the report describing the tools and methodologies for the tasks described in 4.6.1. It includes the Use Case Definition, as well as the study and analysis of end-user requirements for the bedside system related to the current state of the technology. It brings existing but disparate user requirements together and refines them for the UC4.06

2.2 What are expectations of USE-Cases to WP 6.6 bricks

2.2.1 Use Case 2.08

Based on the use-case description in Section 2.1.1, the following requirements were devised up to now:

- Integration of heterogeneous simulation models and executable models
- Ability to integrate new model types and development environments when necessary

- Ability to add meta data to models, and to lookup models when instantiating simulation scenarios
- Ability to link simulation models across companies
- Confidentiality of company specific IP must be assured during simulation

2.2.2 Use Case 4.05: A software centric scalable safety critical medical display platform

We would like to replace the proprietary FPGA-based hardware platform by a COTS platform while keeping the same level of safety and performance.

By changing the design methodology from hardware-centric to software-centric we make it possible to create product variants by simply changing software configurations rather than writing new code or recompiling the code for new target hardware. This will result into a serious reduction in R&D effort spent (30%) on development and maintenance compared to our current hardware-centric platform.

Metrics

To be able to measure improvements, we have defined the following metrics.

1. Traceability of requirements throughout the process:

   For each artifact created during the development process, we can trace back to the corresponding requirement(s).

2. Detection of hardware/software bottlenecks before production/implementation is started:

   The new development process includes model based-engineering tools and techniques to simulate and define the optimal architecture avoiding expensive hardware development/test loops.

3. Consistent ISO 62304 documentation:

   When a product is accepted, a simple push on a button should provide an ISO 62304 compliant document. Having this ability during the product development process is an asset, but not a requirement. This implies that the document generator can talk to all relevant tools, repositories, documents, et cetera to gather all required information.

4. Reuse of components and models:

   Tools supporting component-oriented & modular design allowing product variance based on configuration without the need to recompile code or redesign/recertify modules.

5. Binary reuse of previous FDA approved components (e.g. algorithms):

   Automatic generation of product documentation and product specification for the FDA compliance of a specific product variant reusing a previous FDA approved component.

2.2.2.1 Requirements

Based on the use case description the requirements for the brick from WP 6.6 are listed in Table 2-1. The requirements are split up in functional analysis, design analysis and system level performance simulation.

Table 2-1

Activity	Input	Output	Tool Requirements
Functional Analysis	System Use Cases	Executable Functional Model	Architectural modelling Architectural simulation Traceability between model artifacts to

Activity	Input	Output	Tool Requirements
			System use Cases Model Formal Verification Model base-lining Document Generation Model reuse Model Consistency Checking
Design Synthesis	System Requirements System Use Cases Executable Functional Model	Executable System Architecture: • System Use Cases • Architectural Decomposition • Component Descriptions • Executable Functional Models Traceability of requirements to cases, components, and models	Architectural modelling Architectural simulation Traceability between model artifacts to System use Cases Model Formal Verification Model base-lining Document Generation Model reuse Model Consistency Checking Architectural modelling Performance Modelling and Simulation Architectural base-lining Model and component reuse
System Level Performance Simulation	System Requirements System Use Cases Executable System Architecture	Executable Performance Model Trade off analysis report Performance Analysis Report	Model Consistency Checking

2.2.3 Use Case 4.06: Infusion controller for blood pressure

The system to be developed is very complex and it is intended to control physiological variables of the patient. As the safety of the patient is the first priority in the development, the validation of the system is an essential task.

In the specific Infusion controller scenario, the expectation is to be able to close the loop, using a specific Blood Pressure (BP) monitor, which includes a control algorithm, and computes the adequate dose to be infused. This dose will be sent to a real pump (not SW model) on one side, and also to a SW model of the patient´s behaviour. The result of this SW patient´s model is the change in MAP (Mean Arterial Pressure) as a result of changes of dose infusion values.

The test bench developed to perform Task 6.6.4 – “Integration of Real-Time HiL Testing and Simulation environments” (from project proposal) should provide enough evidence of performance under the most complex conditions.

This simulation tool will provide means to evaluate patient safety requirements by simulating the human body behaviour when vasoactive drugs are injected and give the right response as blood pressure value.

This tool will be used in the validation phase and, from this point of view, its outputs are key for two different processes:

- System Validation
- Compliance with IEC 62304 (Medical Device Software – Software Life Cycle)

- Compliance with legal requirements of the European Union for certification of the system according to medical devices directive 93/42/CEE.

To comply with these objectives it is essential the integration among the tools used in the development process. These tools have been described in 2.2.2. Table 2-2 includes the integration needs that must be covered by IOS. As it can be seen in Table 2-2 the IOS requirements for these tools are related to requirements management or change capabilities to support

- Requirement management,
- Test case management,
- Simulations (of GPU, image processing, controller or human response to medication), and
- Traceability from requirements to test cases.

Table 2-2

Stage	***Integration required***
Requirement Specification	Link Requirements to Architecture and Test Cases
Rapid prototyping of architecture	Link Functional and Performance Analysis and Interoperable Architectural Analysis tools with requirements
Simulation	..
System Validation	Link Requirements and Test Cases. Test cases must be updated with Validation Results. The simulator must be linked with the Test Case to get configuration information and provide results. Link validation results with legal requirements and standard compliance

From a WP6.6 bricks perspective, System Validation is the most important stage in Table 2-2

3 Related work

Simulator coupling is a topic of active research. Scientific literature already documents several couplings that were using specifically tailored interfaces. Additionally, related literature exists with respect to harmonized execution models and coupling frameworks.

One concrete simulator coupling is described in [1]. It illustrates a simulator coupling that was published already in 1997, which integrates the ANSYS and SABER tools for microsystem design. ANSYS is a simulation solution that implements algorithms for solving linear and non-linear engineering problems; SABER is a circuit and system simulator. This simulator coupling was realized for evaluation of thermal properties. It required semantic coupling of two different simulation models, which were realized in these two specialized tools. The authors of [2] describe the coupling of a simulator for netlists and another simulator for Very High Speed Integrated Circuit Hardware Description Language (VHDL) code. By coupling the simulators SLSim and SMASH, the authors enabled integrated simulation of continuous SPICE (Simulation Program with Integrated Circuit Emphasis) simulation models, as well as discrete VHDL models that enable the prediction of electronic circuits.

The work presented in [3] illustrates the networking domain as another application area of simulator coupling. By coupling simulators for link layer protocols and network layers, the authors build an infrastructure for locating interactions between effects on both layers. The work described in [4] applies the simulator coupling approach to the automotive domain for simulating the impacts of Car-to-X systems. This requires coupling of the OMNeT++ simulator, which provides a network simulation, and the road traffic simulator SUMO. The integrated simulation environment is used to evaluate the impact of Car-to-X solutions to vehicle behavior.

These works show that simulator integration is already today a proven technique that is applied whenever the coupling of simulation models becomes necessary. Due to the lack of coupling frameworks, however, development of such a coupling is time consuming, because interfaces and the overall simulation model are created individually for every coupling.

Modelisar [5] is a framework for the coupling of time-triggered simulators. It is based on a common simulation model that realizes time-triggered semantics and data flow communications. Several industry strength tools, for example, Simulink, and Modelica, implement Functional Mockup Interfaces (Modelisar/FMI) as defined by Modelisar. Modelisar, however, is focused on one simulation model; e.g. the integration of event-driven simulators that require simulation of queues is not possible. Therefore, simulation of complex networks, cloud-based services, and non-dataflow models in general are not in the scope of Modelisar.

SystemC/TLM [6] is another approach for the coupling of simulation models that is widely used in the domain of electrical engineering. It supports coupling of time and event triggered processes, but its semantics and simulation model are tailored to the simulation and coupling of components that simulate digital electronic circuits.

The aforementioned frameworks enable the coupling of simulation components, but cover only one simulation model. This limits their scope to one class of simulation areas. Modelisar for example is used in functional design, while SystemC is widely used in platform design. Development of integrated simulation scenarios requires coupling of simulators from different domains that use different simulation models, for example to reflect execution of functional designs on platform models.

Ptolemy II [7] is a framework that is used for exploring the integration of execution and communication models. It splits a system into different domains that are time-triggered, event-triggered, or that describe, for example, the triggering semantics of wireless networks. Each domain realizes one semantic model; Ptolemy defines an approach for the coupling of these domains.

4 Specification of developed solutions

4.1 Approach for integration of heterogeneous models (B2.47)

4.1.1 Brick description

Specialized simulators simulate specific aspects of the real world with high accuracy; other aspects are approximated or ignored. The simulation of complex scenarios requires simulator coupling to address all relevant aspects in one semantically integrated scenario. In the domain of embedded systems, one common example scenario that requires simulator coupling is the simulation of shared control loops.

Today, in automotive, industrial, and avionic environments, control functions that control the same actuators are usually deployed on one electronic control unit (ECU) or computer, which is the hardware the function is executed on. For the near future, plans exist to replace the common units with fewer multi-core CPUs. This way, independent functions for example would be deployed on the same hardware unit, while one function (i.e. one control loop) could be spread over several hardware units as well if this fits the system architecture. This way, the number of required hardware units could be drastically reduced.

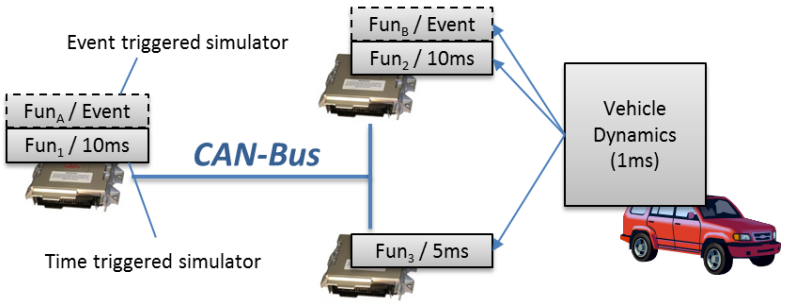

Figure 4-1: Example (automotive) scenario

Figure 4-1 illustrates an example of a highly integrated automotive system architecture. It consists of two control loops, which are deployed onto multiple ECUs. Control loop 1 consists of periodically triggered functions Fun1, Fun2, and Fun3. Control loop 2 consists of event triggered functions FunA and FunB. Event triggered functions are commonly used for the handling of events that require timely processing. Periodic functions are executed at different periods; Fun1 is, for example, triggered every 10ms, while Fun3 is triggered every 5ms. Communication between ECUs is via an automotive network, e.g. via a CAN bus.To simulate all relevant effects, simulator coupling becomes necessary. In the described scenario, this includes the simulation of functional behavior, communication, and vehicle dynamics. Simulating all relevant effects by using specialized simulators separately is not sufficient, because effects often correlate with each other. For example, random delays of input data due to high network load and resulting jitter changes the functional behavior of algorithms. Only the combination of all relevant effects in one integrated holistic simulation scenario reveals all hidden interactions. Complex scenarios therefore need to be simulated by a combination of specialized simulators. Their integration requires the combination of multiple simulation models, which may be time based, event based, or based on finite state machine semantics.

4.1.2 Addressed requirements from Use-Case

This brick mainly addresses the requirements of user story US 2.08:

- Integration of heterogeneous simulation models and executable models

- Ability to integrate new model types and development environments when necessary
- Ability to add meta data to models, and to lookup models when instantiating simulation scenarios
- Ability to link simulation models across companies
- Confidentiality of company specific IP must be assured during simulation

4.1.3 Proposed solution

We propose to solve the challenge of integrating heterogeneous models by coupling individual simulation models through a framework that provides the semantic integration between the models, enables communication according to the Models of Communication and Computation of the individual models and that controls the simulation time, re-synchronization intervals and simulation accuracy of the resulting holistic simulation scenario.

4.1.4 Relationships to dependent bricks

Relationships exist to brick B2.48.

4.2 Simulative evaluation of heterogeneous models (B2.48)

4.2.1 Brick description

Nowadays real products are to be assembled from a broad spectrum of parts obeying a complex set of laws each. On a system and subsystem level these parts of a product mutually interoperate in extensive ways. Models representing such parts (and a combination of such parts respectively) can be used for the simulation of their interplay within controlled system environments thus simulating the behavior of the product to be evaluated. To this end simulating environments take a closed loop control based on events, time, and states to operate an arbitrary and often complex combination of simulation (sub) models.

In homogeneous environments this is realized by implementing the whole model and all its parts in one modeling language, but for different reasons this approach does not fit the needs:

- Parts of the model implementations to be evaluated exist already but may be the intellectual property of another party. Therefore they can't be used until the "wheel is re-invented".
- Already existing models, freely available but implemented in a different language cannot be used and therefore cause double work.
- Limitations of the used simulation environment force for re-modeling of available (sub) models.

Since products' parts are generally built by different producers each and every of the issues mentioned above may be valid for a simulation scenario, making evaluation unnecessarily expensive, and time consuming or even impossible. Therefore a simulation environment is needed that supports the deployment of heterogeneous models for heterogeneous simulation tools.

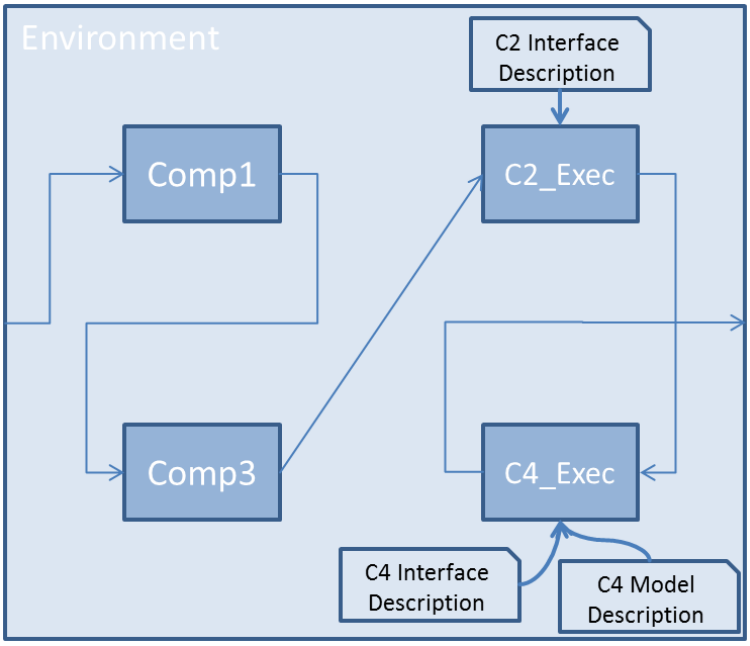

Figure 4-2: Heterogeneous scenario

For the integration of heterogeneous structures into one's simulation environment two principle possibilities can be thought of. On the one hand executable part models must be able to integrate, i.e. when triggered with some input a correspondent output is directly provided (Co-Simulation). On the other hand non-executable part models must be able to integrate by providing a model description that enables the environment to generate an executable (Model-Exchange).

4.2.2 Addressed requirements from Use-Case

This brick mainly addresses the requirements of user story US 2.08:

- Integration of heterogeneous simulation models and executable models
- Ability to integrate new model types and development environments when necessary
- Ability to add meta data to models, and to lookup models when instantiating simulation scenarios

4.2.3 Proposed solution

We propose to solve the challenge of a simulative evaluation of heterogeneous models by deploying model units like functional mock-up units (FMU) within a simulation framework that takes care of the interplay of the different FMUs deployed. By using functional mock-up interfaces (FMI) for co-simulation and model exchange it will be possible to build complex systems consisting of model parts based on a wide variety of simulation tools that already support FMI. While the simulation framework controls the process of deploying the contained heterogeneous models in a closed loop manner based on events, time and states part deployments are delegated to the FMUs that either execute their own simulation (co-simulation) or provide all data for the deployment of their underlying model within the specified simulation environment (model-exchange).

4.2.4 Relationships to dependent bricks

Relationships exist to brick B2.47

4.3 Model-Based Design Verification Method (B2.49)

4.3.1 Brick description

Modelling and simulation of complex systems is at the heart of any modern engineering activity. Engineers strive to predict the behaviour of the system under development in order to get answers to particular questions long before physical prototypes or the actual system are built and can be tested in real life.

An important question is whether a particular system design fulfils or violates requirements that are imposed on the system under development. When developing complex systems, such as spacecraft, aircraft, cars, power plants, or any subsystem of such a system, this question becomes hard to answer simply because the systems are too complex for engineers to be able to create mental models of them. Nowadays it is common to use computer-supported modelling languages to describe complex physical and cyber-physical systems. The situation is different when it comes to describing requirements. Requirements are typically written in natural language. Unfortunately, natural languages fail at being unambiguous, in terms of both syntax and semantics. Automated processing of natural-language requirements is a challenging task which still is too difficult to accomplish via computer for this approach to be of significant use in requirements engineering or verification.

This brick will enhance the vVDR (virtual Verification of System Designs Against System Requirements) method (proposed in [8]) that enables verification of system dynamic behaviour designs against requirements using simulation models. In particular, it will show how natural-language requirements and scenarios are formalized using different tools and exported as FMUs and how simulation models can be composed automatically and used for design verification.

4.3.2 Addressed requirements from Use-Case

This brick mainly addresses the requirements of user story US 2.08:

- Integration of heterogeneous simulation models and executable models
- Ability to integrate new model types and development environments when necessary
- Confidentiality of company specific IP must be assured during simulation

4.3.3 Proposed solution

vVDR artefacts in, such as, requirements, scenarios and design alternative models, can be created in different tools (and also using different formalisms, such as Modelica, UML/SysML, Simulink, etc.). In contrast to the approach presented in [8], this brick will consider the integration of the created executable models will be based on the FMI standard. The challenge will be the question how to capture and resolve dependencies between FMUs in order to enable an automated composition of simulation models for the purpose of design verification.

4.3.4 Relationships to dependent bricks

Relationships exist to brick B2.47

4.4 ModelicaML Tool for Design Verification Based on Models (B2.40)

4.4.1 Brick description

The goal of this brick is to provide modelling and simulation environment for integrating requirement, design alternative and scenario models based on the FMI standard. This implies that models, i.e., FMUs, will be created in different tools (provided each tool used supports the FMI export). The ModelicaML tool will be enhanced to facilitate the usage and integration of FMUs in order to enable model-based design verification (see B2.49).

4.4.2 Addressed requirements from Use-Case

This brick mainly addresses the requirements of user story US 2.08:

- Integration of heterogeneous simulation models and executable models
- Ability to integrate new model types and development environments when necessary
- Confidentiality of company specific IP must be assured during simulation

4.4.3 Proposed solution

We suggest to address this challenge by enhancing the existing ModelicaML tool [9] to allow viewing FMUs data (e.g. public variables) and capturing dependencies between FMUs (see the bindings concept in [8]) in order to enable an automated integration of FMUs for design verification purposes.

4.4.4 Relationships to dependent bricks

The enhanced ModelicaML environment will be used to support the method developed in B2.49.

4.5 Simulation with hardware in the loop capabilities (B4.06)

4.5.1 Brick description

Hardware-in-the-loop (HiL) simulation is a technique that is used in the development and test of complex (real-time) embedded systems. HiL simulation provides an effective platform by adding the complexity of the process under control to the test platform. The complexity of the process under control is included in test and development by adding a mathematical representation of all related dynamic systems. These mathematical representations are referred to as the “process simulation”. The embedded system to be tested interacts with this process simulation.

The goal of this brick is to make HiL interoperable with other engineering methods. We test this by applying HiL to various use cases. To make HiL interoperable, steps that are made should be traceable and there should be interconnection to other tools in the tool chain.

MATLAB is a programming environment for algorithm development, data analysis, visualization, and numerical computation. It is a widely adopted tool for researchers in many fields with an elaborate set of toolboxes and the possibility to create custom functions and models. For hardware in the loop testing, a Matlab model is connected to partial hardware solutions in early stages. Missing functionality is simulated with Matlab (mixed reality configuration). This allows for early analysis of performance and bottlenecks.

4.5.1.1 UC406, An intelligent infusion controller for blood pressure regulation in operating room

In UC406, HiL simulation is used in the system validation step. The device is tested in a simulated environment to be able to perform the tests necessary to obtain certification. See Figure 4-3 for the position of HiL simulation in the development cycle of UC406.

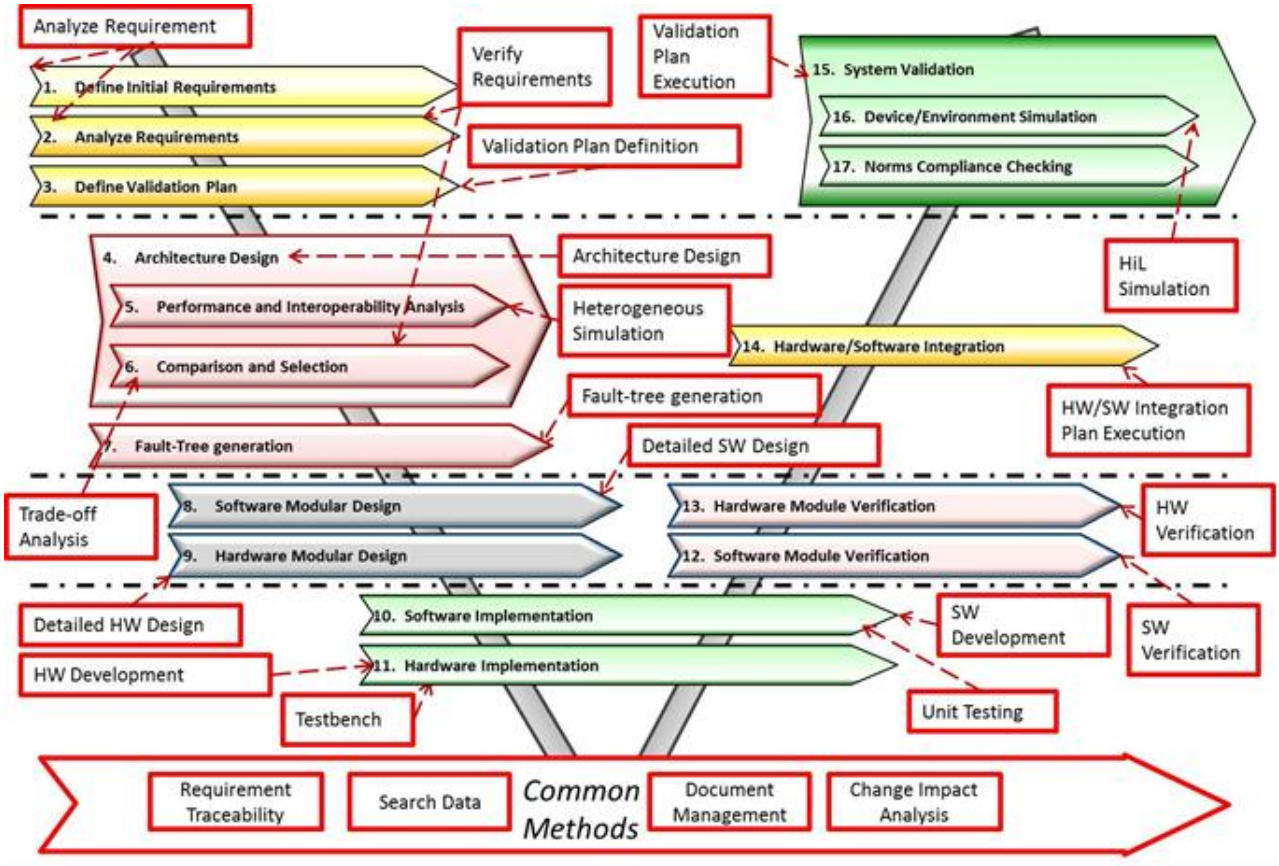

Figure 4-3: Position of various engineering methods in the development cycle, as taken from CRYSTAL_D_406_010_v1_0.doc

From the (verified) requirements, a validation test plan is defined. In the validation test plan experiments are described that can be carried out with the HiL simulation tool. When hardware and software are integrated, one obtains a product that can immediately be implemented and tested in the HiL simulation.

In this specific use-case, the HiL simulation creates an environment to which the intelligent infusion controller can be connected. The HiL simulation should fulfill multiple tasks: it should communicate with the infusion controller in the same way a normal blood pressure sensor and infusion pump do, and it should be able to perform all experiments as described in the validation test plan.

More specific, the main goal to be achieved by this brick for this use case is:

- Simulate the patient's blood pressure evolution based on inputs coming from the Intelligent Infusion Controller.
- The patient's blood pressure is an output value that is used as a input for the Intelligent Infusion Controller

- The behaviour of the Intelligent Infusion Controller when receiving inputs from the patient model taking the appropriate decisions and in the correct timing. In a more precise way this implies that the BP monitor embedded in the product, that includes a control algorithm, computes the adequate dose to be infused at right time and SW algorithms robustness in the presence of noise.
- The correct integration between the BP monitor, (developed by the company) with the infusion pump (provided by a supplier).

Moreover the brick must provide information for the:

- Compliance with IEC 62304 (Medical Device Software – Software Life Cycle) as part of the process to obtain evidences of system validation.
- Compliance with legal requirements of the European Union for certification of the system according to medical devices directive 93/42/CEE.

The requirements for the HiL simulation and for the interoperability between the HiL simulation and the engineering methods before and after it can be found in Table 4-1

Table 4-1: Requirements for HiL simulation and for interoperability between HiL simulation and other engineering methods in UC406

REQ	Description	Stakeholders	Interoperability y/n	MoSCoW - Priority	Rationale
B406.1.	Provide right hardware connections	Certification manager, test engineer	n	M	Provide the right connections for interconnecting the simulator with the product, guaranteeing interoperability between the product to be tested and the simulation environment.
B406.2.	Be able to simulate the correct environment	Certification manager, test engineer	n	M	It must be possible to simulate the environment for the product to confine to the experiments described in the validation plan. The validation plan is set up to conform to the requirements.
B406.3.	Link between an experiment in the validation plan and the experiment that is actually performed.	Certification manager	y	M	It must be possible to keep track of what experiments are done, with which parameters and what the results are. This means that all parameters that need to be changed need to be imported to the simulation from the experiment

REQ	Description	Stakeholders	Interoperability y/n	MoSCoW - Priority	Rationale
					description and that the experiment results need to be exportable.
B406.4.	Comply with norms	Certification manager	n	M	If applicable, the HiL simulation software should comply with certification norms necessary to certify the end product.
B406.5.	Reuse model	System architect, certification manager	n	C	It would be nice if the model that is used within the HiL simulation can also be used in the architecture phase to evaluate the chosen architecture. In this way, the model needs only to be developed and validated once.
B406.6.	Link between system validation and system architecture	System engineer, certification manager	y	C	If the same model is used in system architecture and system validation, there should be a link between the model versions.
B406.7.	Version management simulation	Certification manager, test engineer	y	M	It must be possible to keep track of the version of the simulation version
B406.8.	Version management experiment	Certification manager, test engineer	y	M	It must be possible to keep track of the version of the experiment version, the model version, the simulation version
B406.9.	Version management model	Certification manager, test engineer(, system architect)	y	M	It must be possible to keep track of the version of the experiment version, the model version, the simulation version
B406.10.	Model validation	Certification manager(, system architect)	n	M	The model that is used within the HiL to simulate the environment, in which the product will function, must be validated itself. The validation demands follow from the certification requirements.
B406.11.	Link to model validation tests	Certification manager	y	M	There should be a link between the model validation demands and the model validation tests

REQ	Description	Stakeholders	Interoperability y/n	MoSCoW - Priority	Rationale
B406.12.	Link between hardware and simulation is RS232 or UDP	Test engineer, system architect	y	M	These two communication protocols will be implemented.

4.5.2 Proposed solution

MATLAB is a programming environment for algorithm development, data analysis, visualization, and numerical computation. It is a widely adopted tool for researchers in many fields with an elaborate set of toolboxes and the possibility to create custom functions and models. For hardware in the loop testing, Matlab is connected to partial hardware solutions in early stages. Missing functionality is simulated with Matlab (mixed reality configuration). This allows for early analysis of performance and bottlenecks.

4.5.2.1 UC406, An intelligent infusion controller for blood pressure regulation in operating room

MATLAB will be used as a programming environment. A model simulating (the changes in) the human blood pressure can be made and validated according to the validation plan as defined in the requirements stage. This model can then be used in both the architecture design phase and the system validation stage.

In the validation plan, a set of validation experiments should be defined. The results of these validation experiments are held in the certification document. It should be traceable what version of the product is used, what version of the human model is used and what version of the HiL simulation is used. Also the settings within the experiment should be traceable. See Figure 4-4 for a graphical presentation of the interconnections.

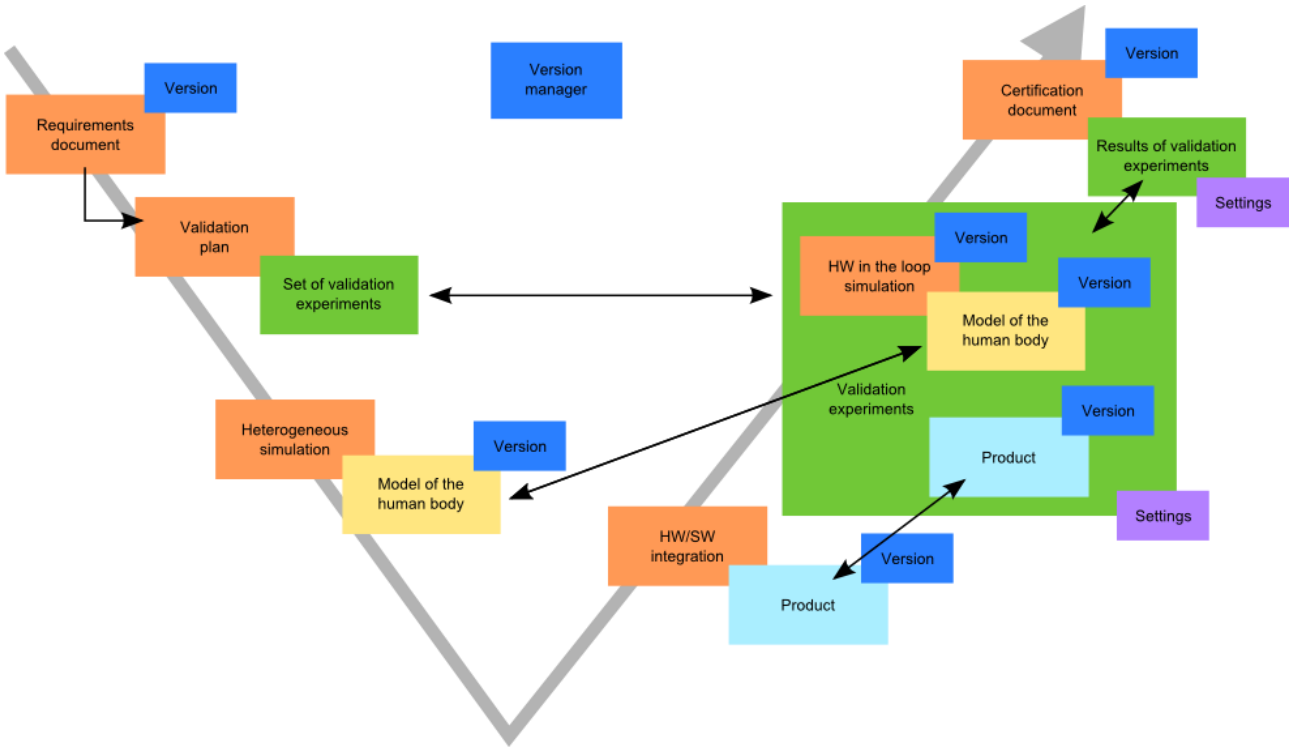

Figure 4-4: graphical representation of the interconnections between the engineering methods, connected to the engineering methods as given in Figure 4-3

4.5.3 Relationships to dependent bricks

Downstream: Validation plan execution

Upstream: HW/SW integration

Sideways: Heterogeneous simulation

Downstream: Crystal bricks (and other tools like Ethercat,)

Upstream: Requirements

4.6 Real-Time HiL for Critical Features (B4.17)

4.6.1 Brick description

The Real-Time HiL brick will provide means for evaluating safety using the HiL concept. In this way this brick provides a basic simulator to be used in a testing environment. The simulator is connected to real hardware to test it under realistic conditions. The simulator provides input to the system under test and reacts to output coming from the system under test.

4.6.2 Addressed requirements from Use-Case

This brick is mainly focused on providing HiL capabilities for the UC406 - An intelligent infusion controller for Blood Pressure regulation in Operating Room. Therefore, the basic and IOS requirements it must fufil are determined by this UC.

In regard to IOS use case needs this brick must be able to support the linking of Requirements and Test Cases with the validation results obtained. The simulator is used in the test case executions.

From these goals and needs detailed in the UC406 derive the requirements listed in Table 4-2. The requirements are classified between the basic and IOS ones.

Table 4-2

Type of Requirement	Title	Description
Basic	Provide right connections	Provide the right connections for interconnecting the simulator with the product, guaranteeing right interoperability between the product to be tested and the simulation environment.
Basic	Model of human	Provide a model of the behaviour of a human blood pressure when doses of a drug are injected
Basic	Noise generation	Provide an algorithm for introducing different noise types and noise intensities in the signal obtained from the patient SW model. Noise is a way of modelling non-usual behaviours in the human.
Basic	Data collection	Collect the data obtained during the simulation and store it.
Basic	Rules setting	Define rules over the collected data in order to detect undesired situations.
Basic	Alert generation	Generate alerts when an undesired situation is detected in the simulation. The detection is based on the rules defined.
Basic	Export data	To support data exportation in common formats (xml, csv…)
IOS	Linking test cases	The tool must support the linking with the Test Cases. The device will be configured according a configuration defined in a particular test case. The link with the requirements is done through the Test Cases.

4.6.3 Proposed solution

Based on the information provided in the previous section, B4.17 must provide the interactions shown in Figure 4-5. The Real-Time HiL is the simulator of one of the system components: a human. The Test Case Manager arranges and executes Test Cases based on its definition. The Test Case Manager is connected to the Real-Time HiL using IOS. The Real-Time HiL, as a device, is connected with the Intelligent Infusion Controller simulating one of the components of the system.

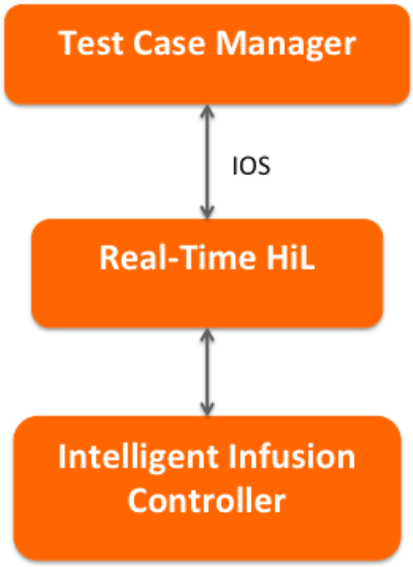

Figure 4-5: Interactions of real-time HiL

Considering the requirements specified in Table 4-2, it has been performed the design depicted in Figure 4-6.

The Real-Time HiL is made of several sub elements:

- Human body simulator: This is the model of the human body (developed in Matlab). It receives the input and calculates the pressure change based on the dose and in other possible conditions.
- Test Configurator: The device is acting as an element in the execution of a Test Case. The configuration, that depends on the test case under execution, must be received.
- Test Executor: Starts and stops the simulation when needed according with the Test Case under execution.
- IOS Layer: Provides the IOS communication service to the other blocks. The IOS allows communication with the Test Configurator and the Test Executor.
- UI: Provides a user interface to use the simulator. It allows communication with the Test Configurator and the Test Executor.

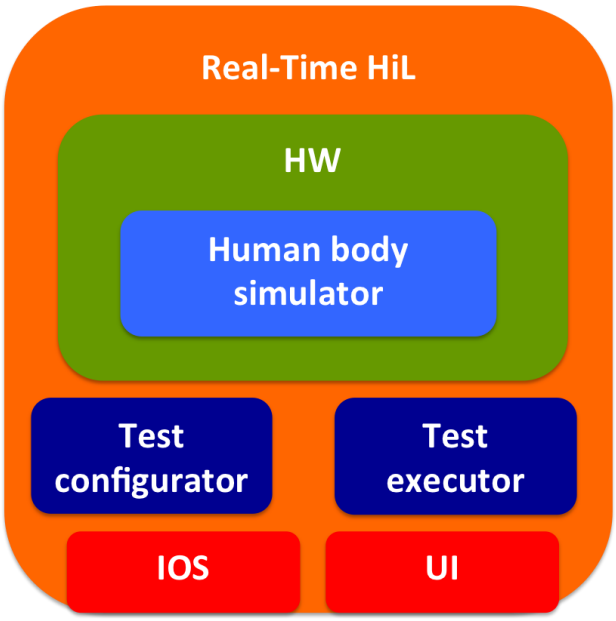

Figure 4-6: Real-time HiL design

4.7 Performance simulation (B4.09)

The Barco use case WP404 was originally viewed as a Hardware in the loop case that might provide context and requirements for brick B4.06 Hardware in the loop. On further analysis and elaboration of the Barco use case WP 404, it appeared that early simulation of architectural options is important, in particular with regards to the image processing pipeline. Since Barco moves to commercial off-the-shelf components, hardware development becomes less prominent. Therefore, simulation of GPU performance for time, spatial, spectral and contrast accuracy is more important than hardware in the loop. For this, B4.09 Performance simulation is foreseen in the Crystal project.

4.7.1 Addressed requirements from Use-Case

For the Barco use case a number of engineering methods have been identified. The identified engineering methods so far are:

Table 4-3

Input	Engineering Method	Output
	Requirements engineering	
Requirements, discrete simulation tool, continuous simulation tool	**Functional modelling**	**Decomposition of model**, executable discrete and/or continuous **model**
	Performance **modelling**	
	Architectural design	
	Architectural trade-off analysis	
	Formal Verification	
	Risk Analysis	

Input	Engineering Method	Output
	Fault Tree Analysis (FTA)	
	Fault Mode and Effects Analysis (FMEA)	
	Certification Reporting	
	Coverage and Impact Analysis	
	Corrective Actions Preventive Actions (CAPA)	
	Software Development	
	Electronics Development	
	Mechanical Development	
	Review	
	Cross Domain Configuration Management	
	Product Variant Management	

For the first CRYSTAL iteration, Barco mainly focusses on the functional modelling. From the use case and defined tool for B2.49, TNO identified the following requirements:

Table 4-4

Requirements to tool	Interoperability	MoSCoW
Link version of decomposition and version of the simulation with settings	**y**	**Must**
Create performance analysis of a possible function decomposition on continuous signals at system level	**n**	**Must**
Create state machine for possible function decomposition on discrete states at system level	**n**	**Must**
Cosimulate discrete and continuous signals	**y**	**Should**
Cosimulate thermal behaviour with image quality behaviour	**y**	**Could**
link version of requirements and version of simulation	**y**	**Must**
link version of test and version of simulation	**y**	**Must**
link version of code and version of simulation	**y**	**Must**

4.7.2 Proposed solution

Model-based design performance verification against requirements approach which enables the detection of inconsistencies or incompleteness of requirements, and allows the engineer to determine which requirements can be verified using simulations. This is possible based on the knowledge which design models are or will be in place. However, the main purpose of vVDR is to facilitate an automated generation of simulation models. This is the process of solving the combinatorial task to select scenarios, that are appropriate to stimulate a given design alternative model, and all requirements that can be verified by running this scenario. Provided such automation it is expected to significantly improve the process efficiency.

4.7.3 Relationships to dependent bricks

Upstream: requirements

Downstream: Decomposition of model

5 Conclusion

5.1 Conclusion

The identified requirements yield new research challenges for heterogeneous simulation scenarios. After the requirements from relevant, individual use-cases have been identified, those requirements will be consolidated and harmonized in the first quarter of 2014. Then, concrete solutions will be developed that address these requirements.

5.2 Terms, Abbreviations and Definitions

Table 5-1: Terms, Abbreviations and Definitions

CRYSTAL	**CR**itical S**YST**em Engineering **A**cce**L**eration
R	Report
P	Prototype
D	Demonstrator
O	Other
PU	Public
PP	Restricted to other program participants (including the JU).
RE	Restricted to a group specified by the consortium (including the JU).
CO	Confidential, only for members of the consortium (including the JU).
WP	Work Package
SP	Subproject
ECU	Electronic Control Unit – Automotive hardware unit that realizes the execution platform for automotive software functions
MOC	Model of computation – the execution model of a simulation model or of any other executable model
MOCC	Model of Communication and Computation – similar to MOC, but also specifies the communication model of the simulation model
vVDR	Virtual Verification of System Designs Against System Requirements

6 References

[1] Wünsche, S., Clauss, C., Schwarz, P., Winkler, F.: Microsystem Design Using Simulator Coupling. In: European Design and Test Conference, 1997. ED&TC 97, IEEE (1997)

[2] Frank, F., Weigel, R.: Co-Simulation of SPICE Netlists and VHDL-AMS Models via a Simulator Interface. In: International Symposium on Signals, Systems and Electronics, 2007. ISSSE '07. (2007)

[3] Siddique, M., Konsgen, A., Gorg, C.: Vertical Coupling Between Network Simulator and IEEE802.11 Based Simulator. In: International Conference on Information and Communication Technology, 2007. ICICT '07. (2007) 127 –130

[4] Schumacher, H., Schack, M., Kurner, T.: Coupling of Simulators for the Investigation of Car-to-X Communication Aspects. In: Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific. (2009) 58 –63

[5] Blochwitz, T., Otter, M., Arnold, M., Bausch, C., Clauß, C., Elmqvist, H., Junghanns, A., Mauss, J., Monteiro, M., Neidhold, T., Neumerkel, D., Olsson, H., Peetz, J., Wolf, S.: The Functional Mockup Interface for Tool independent Exchange of Simulation Models. In: Proceedings of the 8th International Modelica Conference. 8th International Modelica Conference, 20.-22 März 2011, Dresden.

[6] Ghenassia, F. (Editor), Transaction-Level Modeling with Systemc: Tlm Concepts and Applications for Embedded Systems, Springer 2006.

[7] Eker, J., Janneck, J., Lee, E., Liu, J., Liu, X., Ludvig, J., Sachs, S., Xiong, Y.: Taming heterogeneity - the Ptolemy approach, Proceedings of the IEEE, 91(1):127-144, January 2003.

[8] Schamai, W. (2013). Model-Based Verification of Dynamic System Behavior against Requirements: Method, Language, and Tool.Ph.D. thesis.

[9] ModelicaML – A UML Profile for Modelica. http://modelicaml.openmodelica.org/