 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 Definition
UC 4.6
An intelligent infusion controller for Blood Pressure regulation in Operating Room
D406.010

DOCUMENT INFORMATION

Project	CRYSTAL
Grant Agreement No.	ARTEMIS-2012-1-332830
Deliverable Title	Use – Case Definition
Deliverable No.	D406.010
Dissemination Level	CO
Confidentiality	R
Document Version	V2.0
Date	2014-04-30
Contact	Eduardo Lluna (elluna@iti.es)
Organization	ITI
Phone	
E-Mail	

AUTHORS TABLE

Name	Company	E-Mail
Eduardo Lluna	ITI	elluna@iti.es
Ricardo Ruíz	RGB	rruiz@rgb-medical.com

CHANGE HISTORY

Version	Date	Reason for Change	Pages Affected
1.01	24/10/13	First version	All
1.02	19/11/13	Changes after a revision	7-18
1.03	25/02/14	Expanded the engineering method section.	All
2.0	22/04/14	Added corrections and new sections	All

CONTENT

Content of Tables

Content of Figures

1 Introduction

1.1 Role of deliverable

This document has the following major purposes:

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

1.2 Structure of this document

The document is structured as follows:

- Section 2 presents the use case description at high level.
- Section 3 presents a detailed description of the use case.
- Section 4 presents the engineering methods identified in the use case.
- Section 5 shows the terms and abbreviations used.
- Section 6 presents the references.
- And finally there are annexes including the detailed description of the use cases and the technology base line.

2 High level description of use case and content

2.1 Use case description

Use Case 4.6 presents the development of a medical device. The main challenge in the development process consists on obtaining a product that provides the functionality and meets the certifications and norms required for its use in real medical environment. The fact of fulfilling these norms forces the development process to accomplish some required steps and tasks for providing evidences about the correct behavior of the system and that the development process has been performed as expected.

This use case presents an additional challenge because the company is moving from a development cycle not supporting all the evidences required to certify a product to a one able to generate such evidences. The inclusion of some new steps and tools to support new engineering methods is required. This change is due to the fact that some regulations have changed and because the new product must be designed to a higher safety level compared with previous products developed by the company.

The system under development is a drug infusion device that delivers vasoactive drugs to maintain the patient blood pressure under some limits in Operating Rooms (OR) or Intensive Care Unit (ICU). The system operates as a closed control loop reading the blood pressure and applying the required drug quantity to reduce it, if required. A special algorithm based on fuzzy logic is used for performing such monitoring and controlling.

Figure 2-1 shows the components of the system. The device to be developed is labeled as Control Algorithm. The perfusor (element that injects the drug) and the blood pressure measurer are COTS devices. Figure 2-2 shows real COTS components used in this product.

In general the device consists of the following components:

- A multi-parameters vital sign monitor. The device that displays information and allows the interaction with the device.
- An infusion pump. Injects the drug to the patient.
- Communication bus between components.
- Controller device that executes the control algorithm.

The development of the system covers both hardware and software.

The main benefits of this product, among others, are:

- Diagnosis and therapeutic capabilities
- Enhancing patient care
- Releasing the nursing work

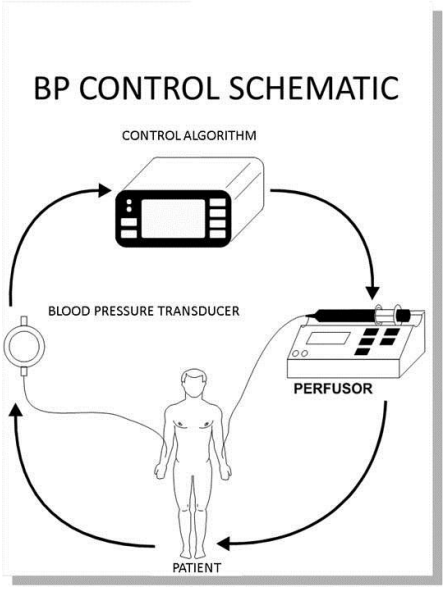

Figure 2-1: System components diagram

Figure 2-2: Real device

2.2 Detailed system behaviour

The system delivers vasoactive drugs with the ultimate goal of reducing patient's hypertension, and precisely controlling blood pressure measurements in a patient undergoing surgical intervention or in post cardiac surgery in Intensive Care Unit (ICU).

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 in use. 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 high and the project would contribute to improve working quality of the clinical staff and the patients' safety.

The device 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 equally demanding areas.

In a typical scenario, the physician in charge will define, at the beginning of the process, the target MAP (Mean Arterial Pressure) where he/she considers the patient is adequately controlled. Vasodilator drug will be infused continually; every 20 seconds the system will decide the new infusion value applying a control algorithm. This will take into consideration past behavior of patient's MAP to drug infusion.

For this purpose, the system needs to integrate several features at the component level:

- Means to measure MAP values, from an either invasive or no-invasive Vital Signs Monitor (VSM)
- Means to infuse vasodilator drug from one or more Infusion Pumps. This accessory must be fully integrated and interoperable with VSM.
- Safe communication between above components.
- A control algorithm
- Connectivity means to an Information System

Then thanks to the integration of these features the product will be able to control the delivery of vasoactive drugs through infusion and monitor the effects on the patient in order to guarantee his/her wealth.

2.3 Detailed description of the Process

The development process is based on a V cycle as shown in Figure 2-3. This use case is mainly centered on the tasks at the top of the V, namely, the Requirement Specification (1), Rapid Prototyping of the Architectural Design (2) and the System Validation (7). This is due to the fact that those tasks provide most of the information required as evidences for the certification process and, therefore, the automation and the interconnection of such a data will produce a high benefit in the development cycle.

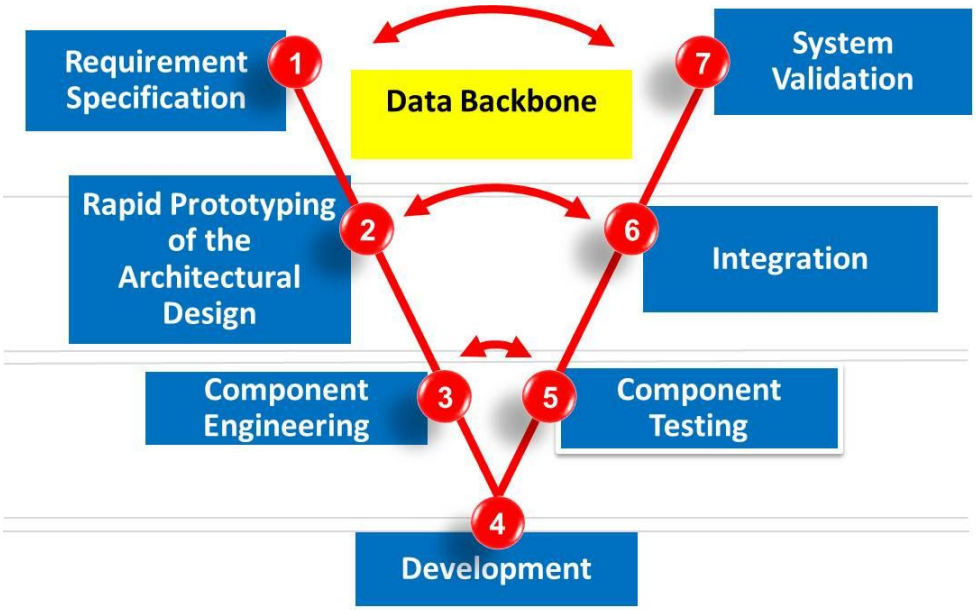

Figure 2-3: Development Cycle

The objective is to have a connection between Requirements Specification and the Architecture Design, to allow a more efficient design cycle and a connection between Requirement Specification and System Validation. The main issue is to trace requirements to design elements and to validation test cases. This is a very important task and a must for the certification process.

As previously mentioned, the company is moving from a very basic development cycle, that is shown in Figure 2-4, to this new model and with the incorporation of some tools to speed up the process.

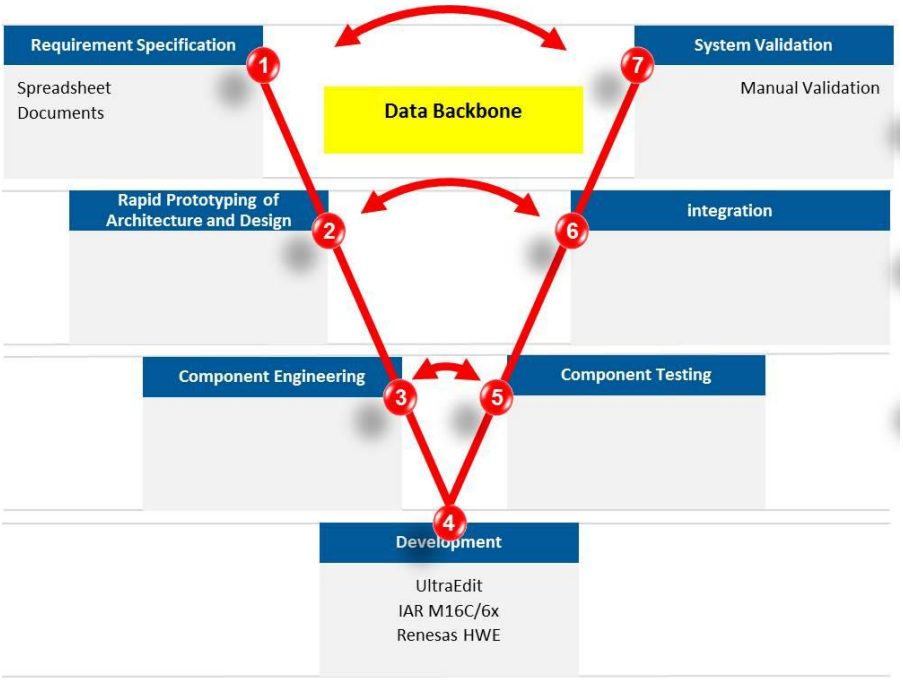

Figure 2-4: Current development cycle

The new development process starts specifying the requirements and a Requirement Management tool is used. The requirements are stored in a database. From the interoperability point of view, for medical devices there are some standards that must be taken into consideration, therefore a set of interoperability requirements must be added to the product requirements. Those requirements come from the B4.16.

The design of the system is based on the requirements. In the design process there are some tools to carry out performance and interoperability analysis to help in the task of choosing the best architectural design. Those tools are the ones developed in B4.14 and B4.15 for performance and interoperability analysis.

The validation test cases can be created once the requirements are available. The test cases must be linked with the requirements. A Test Management tool is required for storing the test case information and for the traceability needs.

In the Validation process some simulators are often needed instead of a real device or environment because availability problems that prevents of using a real device or environment. The validation process includes the use of Hardware in the Loop (HiL) device developed in B4.06.

The HiL developed in the B4.06 includes a model of the human behavior. This modelling is limited to patient´s behaviour when vasodilator drugs are infused and therefore it is restricted to the cardiovascular system, and it relates changes in Mean Blood Pressure to Dose Infusion. In practice, it is implemented by a transfer function where some parameters can be modified in order to modify patient´s behaviour.

The reason to do it is to be able to test the equipment performance in Real Time, but without the need to do it with real patients. It is considered as a preliminary stage, since outside the scope of the CRYSTAL project, it is expected to perform Clinical Evaluation as part of the development process, and later for product certification.

It is clear that there are integrations needed between the tools used in different phases of the development process. There is a clear link between stage 1 and 7, and 1 and 2. These connections define the interoperability requirements for this use case.

Figure 2-5 shows the new development cycle including the new tools that are going to be integrated and that require interoperability among them.

In the following sections the tools and its function in the development process are shown in detail.

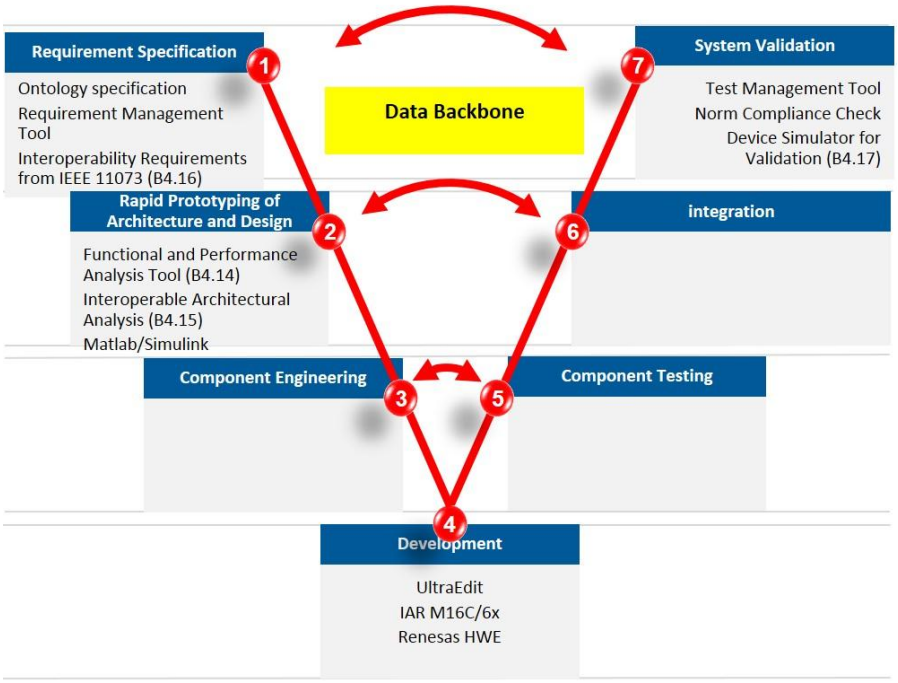

Figure 2-5: Detailed IOS challenges

3 Description of the use case process

3.1 Set of proposed process activities

The activities carried out in each phase of the development cycle are shown in Figure 3-1.

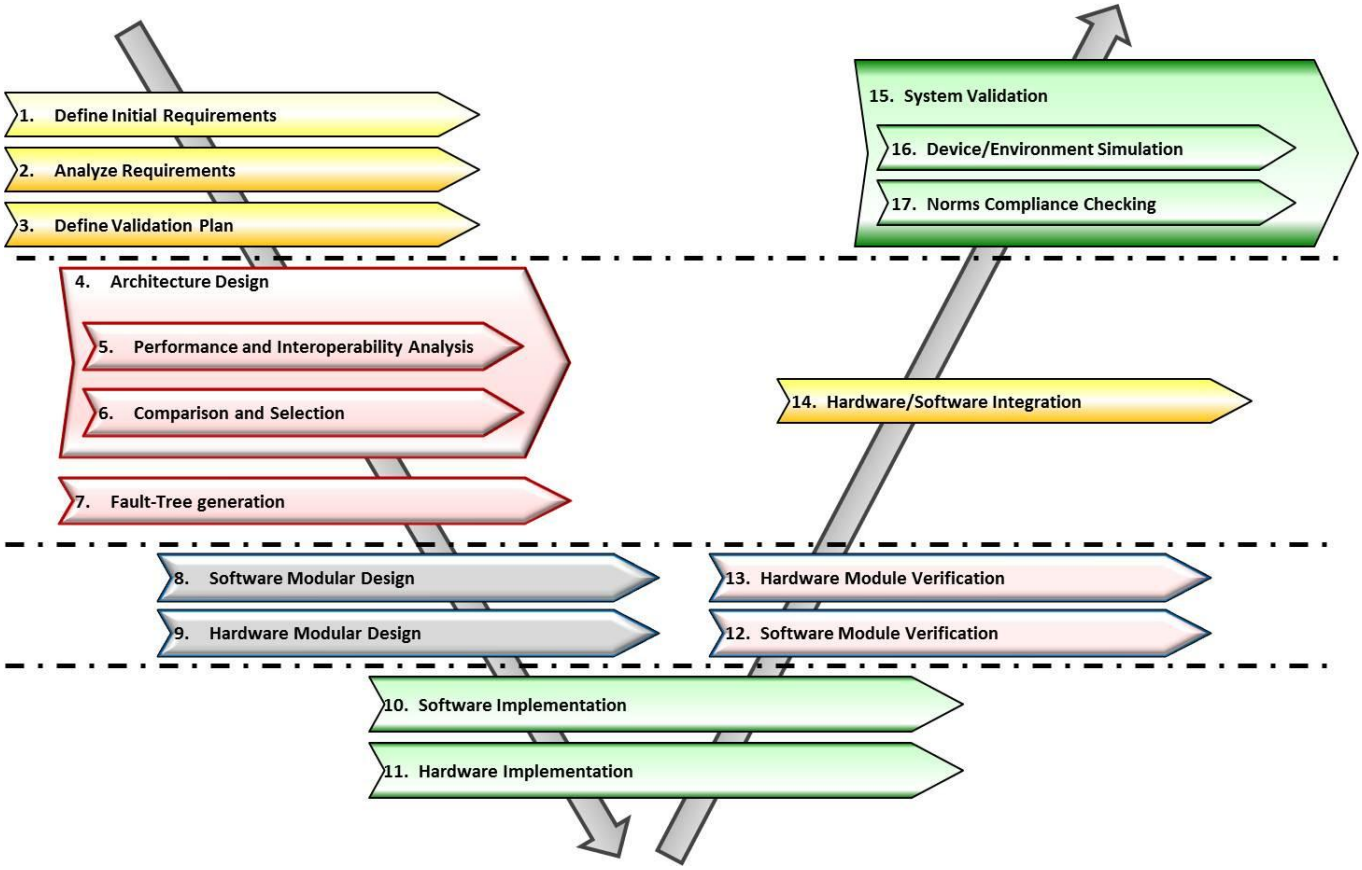

Figure 3-1: Detailed activities in the development cycle

Moving in the V cycle the process starts in the Requirements Specification (1) that consists in the definition of the stakeholder requirements and the analysis of the requirements (2) to eliminate duplicities or unfeasible ones and obtaining the detailed requirements. When the requirements are clear, the Validation Plan (3) can be defined based on the functionality expected for the system.

Next step is the Architectural Design (4). The system is made out of hardware and software components and must be defined which functions are provided by hardware and software. An important task is to check that the defined architecture really meets all requirements (5). Some tools can be used for that purpose (performance and interoperability analysis tools). In some cases possible architectures can be defined and one must be finally selected (6). A detailed requirement list for each of the architecture elements (hardware and software) is extracted from this phase.

Developing safety critical systems a fault-tree based on the architecture must be generated (7).

Now the design must be split in parallel processes, one for each engineering discipline being (hardware, software, mechanics, etc). A more detailed design based on modules is generated for both hardware and software (8)(9).

The software and hardware development is based on a modular design (10)(11). Modular development speeds up the process because it allows several teams to work in parallel in different modules.

When the development is finished every module must be verified (12)(13) and then the integration process (14) must take place. Typically there are two integration processes: software integration and system integration. Software integration checks that the different software modules work together and finally, the system integration checks that all the system components of all engineering disciplines work together.

The final step in the development process is the Validation (15). The validation tests are defined at the beginning of the process based on the requirements. The validation will use the complete system. In some cases not all devices or the real environment are available and some simulators are needed (16). In the present use case, a patient is needed to validate the system. Using a real human has ethical implications therefore a human simulator is used instead for validation before starting clinical tests with real patients.

Norm compliance checking (17) is also important because the final product must be certified according to the applicable norms.

The Table 1 shows a summary of the process activities identified. The table includes the name, a description and a column indicating whether a tool can be used or not.

Id	Activity	Description
1	Define requirements	Based on customer, market and certification requirements a basic product requirements must be produced
2	Analyze requirement quality	The requirements must be analyzed to check that are feasible, not contradictory and really fulfill the customer/product expectations. Detailed requirements are produced from stakeholder requirements. Traceability with stakeholder requirements must be maintained.
3	Define validation plan	Based on the requirements the validation plan can be defined. It must include all test cases that will test all the requirements. Scenarios and required equipment must also be detailed. Traceability is created from the Detailed Requirements to the Test Cases.
4	Architecture design	Define the architecture of the system (hardware/software decomposition, etc.) to fulfill the requirements. Traceability is created from the architectural components to the Detailed Requirements. Detailed requirements for components are generated.
5	Performance and interoperability analysis	Performance and interoperability is dependent on architecture, therefore different possible architectures can be evaluated to select the best one for the requirements fulfillment.

6	Comparison and selection	The most suitable architecture, among the possible, is chosen.
7	Fault-tree generation	A safety critical system must define a fault-tree to help in the definition of the safety requirements.
8	Software modular design	Detailed design of the software based on the decomposition into modules. Traceability is created from the architectural components to the detailed design components.
9	Electronic modular design	Detailed design of the hardware based on the decomposition into modules. Traceability is created from the architectural components to the detailed design components.
10	Software implementation	Implementation of the software source code. Different approaches, techniques and tools can be used.
11	Hardware implementation	Implementation of the hardware. Different approaches, techniques and tools can be used.
12	Software module verification	After the implementation each software module must be verified to check that it behaves as specified
13	Hardware module verification	After the implementation each hardware module must be verified to check that it behaves as specified.
14	Hardware/Software integration	It must be checked that the software runs with the hardware.
15	System Validation	System validation based on the specifications. The validation tests defined at the beginning of the project are executed. The plan defines all the tests, the pass/fail conditions and so on.
16	Device/Environment simulation	In the system validation some devices or the environment can be simulated if the real ones are not easily available and there are some risks. Some simulators can be developed to emulate those devices or environments.
17	Norms compliance checking	If the final product is going to be certified under a norm, the compliance with the norm must be checked.

Table 1: Process Activities identified

4 Identification of Engineering Methods

4.1 Engineering methods

From the activities seen in the previous section (Figure 3-1) the related engineering methods are extracted. An engineering method describes how an activity can be conducted using guidelines, tools and languages that interoperate with each other.

Figure 4-1 shows the engineering methods detected in the development cycle. Each one is framed with a red box and an arrow points to the activity it is related with.

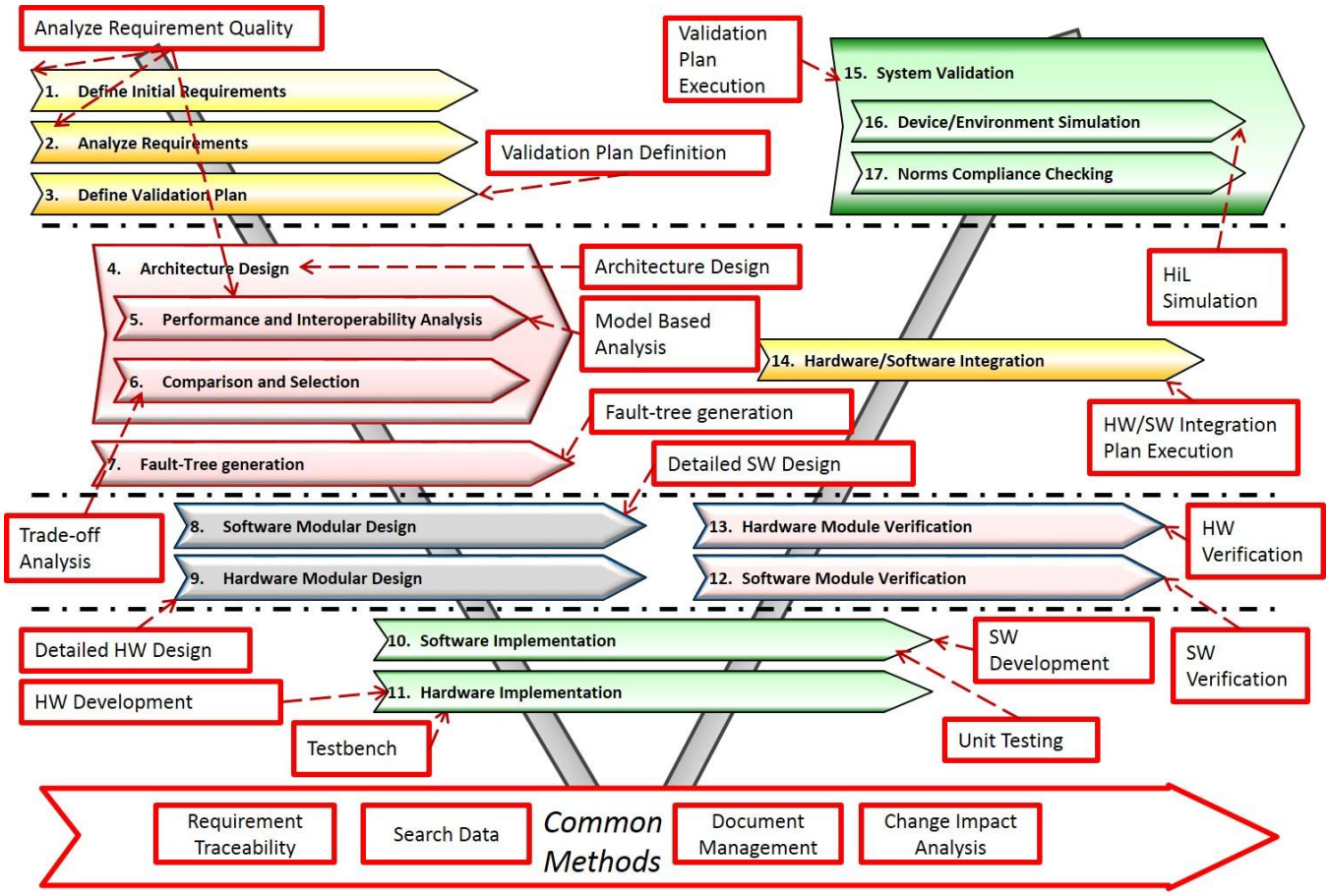

Figure 4-1: Identification of Engineering Methods

The Table 2 shows a summary of the detected engineering methods and a short description summarizing their function in the V model of the use case.

Engineering method	Description
Analyze Requirements Quality	Consists on all the activities to analyze all the requirement collected to check the completeness and that they really provide all the required functionality for all stakeholders.

Validation Plan Definition	Based on the requirements consists on the preparation of the validation process by the definition of the test cases, the environment, equipment required and so on. It is very important the traceability of the tests cases with the requirements. Test cases must assure full requirement coverage.
Architecture Design	All processes and tasks required for defining the architecture for the system that fulfills all the requirements.
Model Based Analysis	In the process of the architecture design allows the creation of system models to check whether the architecture meets the requirements. The process includes the possibility of creating multiple possible model architectures and select the most suited for the problem. Possible analysis are: timing, throughput, interoperability, etc.
Trade-off Analysis	Process of electing among alternatives based on criteria previously defined.
Fault-tree generation	Creation of a fault-tree for the system. This is a critical task for safety-critical applications and depends completely on the product domain. It includes the Fault Tree Anallysis.
Detailed Software Design	Activities to decompose the software into modules that have very little coupling among them (and a clear interface) that can be developed independently.
Detailed Hardware Design	Activities to decompose the hardware into modules that have very little coupling among them (and a clear interface) that can be developed independently.
Software Development	Development of the source code. There are many possible techniques, processes and tools.
Hardware Development	Development of hardware. There are many possible techniques, processes and tools.
Unit Testing	Unit testing consists on the test of the small pieces of code, as they are being developed to assure that we build over solid basements. There are several possible tools and strategies that mainly depend on the programming language.
Testbench	Similar to unit testing but used in hardware development. Consists on the execution of the different hardware blocks simulating the inputs to check the output.
Software Verification	Process of verification of the software. There are many different techniques and tools and the election of the ones to use depends on the safety level required for the final application. Basic techniques applied are static code analysis and dynamic code analysis,
Hardware Verification	Process of verification of the hardware. There are many different techniques and tools and the election of the ones to use depends on the safety level required for the final application.
Hardware/Software Integration Plan Execution	The integration of the hardware and software developed separately.
HiL Simulation	The process of defining and developing simulators to be used instead of some peripheral or the environment. Simulators are very important elements because in

	complex system not all the peripheral or a real environment can be used on testing.
Validation Plan Execution	Execution process of the validation as specified in a plan. The plan includes the definition of the test cases, the test environment and checking the result.
Requirement Traceability	The requirements fulfillment must be traceable through the architectural elements and the test cases. This is very important to help in impact analysis and to evaluate the implication in case of failure.
Search Data	Management of all the data generated in the projects to help in the process of generating documental evidences.
Document Management	Management of all documents generated in the process. It is important to control versions, document status and life cycle.
Impact Analysis	In case of changes it is necessary to be able to determine the scope of such a change.

Table 2: Engineering methods

4.2 Engineering methods details

The following sub sections explains in detail each of the engineering methods shown in Table 2.

4.2.1 Analyze Requirements Quality

The engineering method consists on analyzing the set of requirements to check its quality. Quality means that they really represents the required functionality for all the stakeholders. Several factors can be checked like completeness, formal aspects, etc.

Input	Requirements from all stakeholders
Output	Detailed Requirements verified
Tools	Requirement Management tool
Interoperability	No interoperability requirements

4.2.2 Validation Plan Definition

Based on the requirements the engineering method consists on the preparation of the process required to check that the requirements are met. That include the definition of the test cases, test environment, test equipment, pass/fail criteria and so on.

Test cases must be linked with the requirements. Traceability is very important.

Input	Detailed Requirements
Output	Validation plan (set of use cases)
Tools	Test Management tool
Interoperability	Connection with the Requirement Management Tool

4.2.3 Architecture Design

This engineering method is the process to create the component architecture that meets the requirements.

Input	Requirements
Output	Architecture Design
Tools	Tools developed in B4.14 and B4.15
Interoperability	Connection with the Requirement Management tool.

4.2.4 Model Based Analysis

In the process of architecture design, the Model Based Analysis allows to check the suitability of an architecture to meet the requirements based on the creation of a model. The model allows to perform different kind of analysis such of timing, throughput, interoperability, etc.

Models are much easier to build than the final device, so several different architectures can be modeled as a solution for a problem. A trade-off analysis can be carried out by comparison between models and the architecture that best meets the requirement can be selected.

Input	Detailed Requirements
Output	Architecture Design validated
Tools	Model Based Analysis Tools
Interoperability	Connection with Requirement Management tool

4.2.5 Fault-tree generation

The Engineering Method consists in the creation of the fault-tree for the system. This is a critical task for safety-critical applications and depends on the product domain.

Input	Architecture Design
Output	Fault-tree
Tools	No tool used in this UC
Interoperability	No tools involved

4.2.6 Detailed Software Design

The Engineering Method include the activities required to decompose the software into modules that have very little coupling among them (and a clear interface) that can be developed independently.

Input	Architecture Design
Output	Detailed Software Design
Tools	No tool used in this UC
Interoperability	No tools involved

4.2.7 Detailed Hardware Design

The Engineering Method include the activities to decompose the hardware into modules that have very little coupling among them (and a clear interface) that can be developed independently.

Input	Architecture Design
Output	Detailed Hardware Design
Tools	No tool used in this UC
Interoperability	No tools involved

4.2.8 Software Development

The Engineering Method covers the process to develop the source code. There are many possible techniques, processes and tools.

Input	Detailed Software Design
Output	Software Source Code
Tools	Development tools that depend on programming languages (compilers, debuggers, etc)
Interoperability	No required in this UC

4.2.9 Hardware Development

The Engineering Method covers the process to develop the hardware components. There are many possible techniques, processes and tools.

Input	Detailed Hardware Design
Output	Hardware Schematics or source code in a hardware description language (HDL)
Tools	Development tools that depend on technologies (VHDL compiler, Simulators, Schematics design, PCB design, etc)
Interoperability	No required in this UC

4.2.10 Software Verification

The Engineering Method is the process of verifying that the software developed is according the specifications.

There are many different techniques and tools and the election of the ones to use depends on the safety level required for the final application. A basic technique is the Unit Testing that consists on testing all the small pieces of code, as they are being developed.

The Software Verification is included in the Software Development process.

Input	Verification Plan
Output	Software Verified
Tools	Several possible tools depending on languages, technologies and safety levels. Basically is based on static code analysis and dynamic code analysis. Tools like PolySpace, Understand C, and so on can be used.

Interoperability	No required in this UC

4.2.11 Hardware Verification

The Engineering Method is the process of verifying that the hardware developed is according the specifications.

Input	Verification Plan
Output	Hardware Verified
Tools	Several possible tools depending on technologies and safety level. The development is mainly based on microcontrollers.
Interoperability	No required in this UC

4.2.12 Hardware/Software Integration Plan Execution

The integration between Hardware and Software consists on checking that the developed software runs in the developed hardware.

Input	Integration Plan
Output	Software integrated in the hardware
Tools	No tools used in this UC
Interoperability	No tools involved

4.2.13 HiL Simulation

The Engineering Method consists in the development of simulation that operate in real time and substitutes a component of the system or the environment. They are a very important part because in complex embedded systems not all the components or the real use environment are available during the development.

Input	HiL specifications based on test cases
Output	HiL device
Tools	No tools used in this UC
Interoperability	No tools involved

4.2.14 Validation Plan Execution

The Engineering Method consists in the execution of the validation process as defined in the validation plan. The plan includes the tests and all the details required to the correct execution of the tests. In some cases

Input	Validation Plan
Output	Test cases results
Tools	Test Case Management tool, Test Case Execution tool, HiL device
Interoperability	Connection between Test Case Management tool and HiL device.

4.2.15 Requirement Traceability

The requirements must be traceable with the architecture components and until the test cases. The Engineering Method consists in the creation of the traceability matrix linking requirements with architecture elements and test cases.

Input	Requirements, Architecture Components, Test Cases
Output	Traceability Matrix
Tools	Requirement Management Tool
Interoperability	Connection between Requirement Management tool and Test Case Management tool.

4.2.16 Document Management

The Engineering Method is related with the management of the documentation generated in the process. It covers the version control, document status and life cycle.

Input	All documents generated in the project
Output	Documents properly stored and organized
Tools	Version control system and backup system.
Interoperability	No tools involved

4.2.17 Impact Analysis

The Engineering Method covers the process of determining the parts of the project affected by a change. It is based on the traceability matrix.

Input	Change Proposal, Traceability matrix
Output	Identification of the elements affected by the change
Tools	Requirement Management tool, Test Case Management tool
Interoperability	Connection between Requirement Management tool and Test Case Management tool

4.3 Traceability between user needs, engineering methods and tools:

Next step is the identification of the appropriate tools to be used in each of the engineering methods. As previously mentioned, this use case includes the incorporation of new tools to the development cycle and a selection process was carried out to select the tools.

The Table 3 shows the relation between the processes, the associated engineering method and the tools involved. The tools to be used have been selected after an evaluation process.

Number	Process (= Set of Activities)	Engineering Methods	Tools Involved
1	Define Initial Requirements	Analyze Requirements	IBM DOORS NG
2	Analyze Requirements	Analyze Requirements	IBM DOORS NG

		Verify Requirements	
3	Define Validation Plan	Validation Plan Definition	IBM Quality Manager
4	Architecture Design	Architecture Design	Matlab/Simulink
5	Performance and Interoperability Analysis	Heterogeneous Simulations Trade-off Analysis	Matlab/Simulink, B4.14, B4.15
6	Arch. & Design. Comparison and Selection	Trade-off Analysis	No tool
7	Fault-tree generation	Fault-tree generation	No tool
8	Software Modular Design	Detailed Software Design	No tool
9	Hardware Modular Design	Detailed Hardware Design	Pspice
10	Software Implementation	Software Development Unit Testing	UltraEdit, IAR M16C/6x, Renesas HWE, PCLint
11	Hardware Implementation	Hardware Development Test Benches	Protel Design Explorer
12	Software Module Verification	Software Verification	No tool
13	Hardware Module Verification	Hardware Verification	No tool
14	Hardware/Software Integration	Hardware/Software Integration Plan Execution	No tool
15	System Validation	Validation Plan Execution	IBM Quality Manager, B4.06
16	Device/Environment Simulation	HiL Simulation	Matlab/Simulink, B4.06
17	Norms Compliance Checking	Check norms compliance	No tool
CM-2	Search Data	Search Data	No tool
CM-3	Document Management	Document Management	No tool
CM-4	Change Impact Analysis	Impact Analysis	IBM DOORS NG

Table 3: Process Activities mapped on Engineering Methods

4.4 Interoperability requirement between tools

As previously seen, there are relations among the tools. These relations define the interoperability requirements. Table 4 summarizes the interoperability relations.

Tool	Talks to	Direction	OSLC Specification
IBM DOORS NG	IBM Quality Manager	Bidirectional	Core, Requirements
IBM Quality Manager	HiL (B4.06)	Bidirectional	Core, Quality
Performance Analysis (B4.14)	IBM DOORS NG	Bidirectional	Core, Requirements
Interoperability Analysis (B4.15)	IBM DOORS NG	Bidirectional	Core, Requirements

Table 4: Interoperability between tools

4.5 Engineering Method Development

Not all the Engineering Methods identified in the development cycle are going to be developed in detail in the scope of the project. In this use case the development is centered in those Engineering Methods that are affected by the integration of a new tool or those that provide some interoperability requirement, as well as those that contribute to the certification process required for the final product.

In particular the use cases that are going to be developed in detail are the following:

- Model Based Analysis
- Validation Plan Definition
- Validation Plan Execution
- Requirement Traceability
- Impact Analysis

The Annex 7 include detailed information about these engineering methods.

The following section describes in detail the demonstration for each of the selected engineering methods.

4.6 Demonstrators Description

4.6.1 Demonstration Engineering Method UC406_Validation_Plan_Execution_003

4.6.1.1 Objective

The objective is the execution of the Engineering Method in a real environment using the tools required and obtaining a result.

The demonstration of the Engineering Method is decomposed in several scenarios. Each scenario performs a specific action that is needed to perform the EM.

4.6.1.2 Tools used

Figure 4-2 shows the tools used in the Engineering Method and their relations.

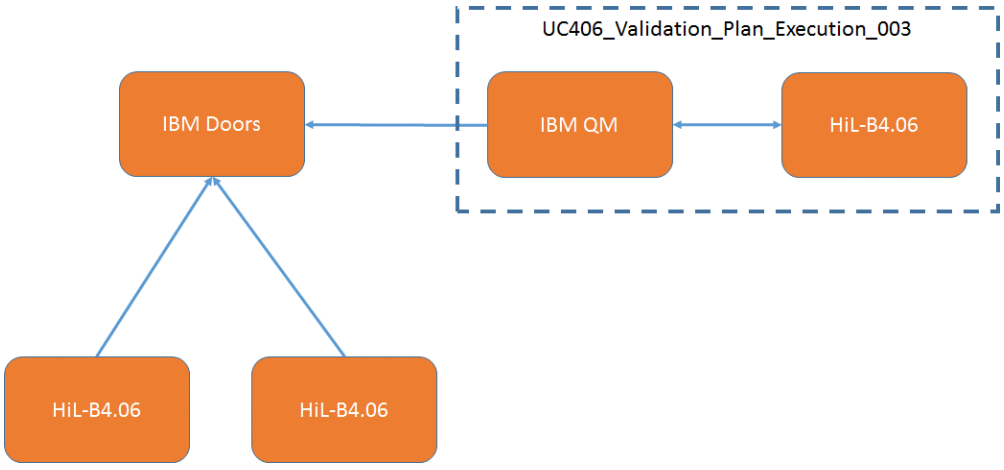

Figure 4-2: Tools involved in the EM

The following table shows the detailed information of the tools used.

Tool Type	Tool Name	Version	Vendor
Test Management	Quality Manager (QM)	4.0.5	IBM
HiL Device	HiL – B4.06	1.0	ITI/TNO

The Test Plan must contain a list of Test Cases.

The Test Cases stored in the QM are used by the HiL-B4.06 to validate the correct functionality of the controller, so they must include the information required to configure start-up of the controller and HiL-B4.06 behaviour.

In order to configure start-up controller, the following information is required:

- Target value of controlled variable (pressure)
- Initial dose of infused drug

In order to define HiL-B4.06 behaviour the next information is required:

- Patient model type
- Patient model parameters: SNP time constant, pulmonary flow time constant, systemic time constant, patient gain and others. These parameters are related with modeling changes of patient pressure with infused drug
- Other configuration parameters: Initial pressure of the patient
- Noise profile. Noise is a way of modelling unusual behaviours in the patient model.

According to previous indications, the Test Cases stored in the QM used by the HiL-B4.06 must include, among other general fields, the following ones:

Feld Name	Type	Description
HiL_required	Boolean	Informs if a HiL-B406 device is required for test execution
Execution_time	Time	How long the test execution lasts
Auto_simulation	Boolean	The simulation will run automatically and will start and stop without external intervention
PASS_FAIL_condition	Integer	The test is passed if patient pressure has been in a well controlled range during a percentage of time greater than a given value (so values of pressure and infusion dose should be recorded during the test in order to be able to calculate percentage of time in a well controlled range).
PASS_FAIL_level	Integer	Level used for determining if test result is positive or negative.
Non_Invasive_Pressure _Type	Boolean	Informs if pressure monitoring is non invasive (discontinuous) or invasive (continuous)
Controller_Target_Pressure	Integer	Target value of controlled variable (pressure) that is set for the user to start the control
Controller_Initial_Dose	Integer	Initial dose rate that is set for the user to start the control
Patient_Model_Type	Integer	Identification of model type used by HiL-B406 device
Initial_Patient_Pressure	Integer	Initial condition of patient pressure for the patient model
SNP_Time_Constant	Integer	Configurable parameter of patient model
Pulmonary_Flow_Time_Con stant	Integer	Configurable parameter of patient model
Systemic_Time_Constant	Integer	Configurable parameter of patient model
Patient_Gain	Float	Configurable parameter of patient model

Noise_Profile	Integer	Characterization of noise introduced by HiL-B4.06 device

The following images Figure 4-3 and Figure 4-4 show some captures of QM as it would be used for the engineering method.

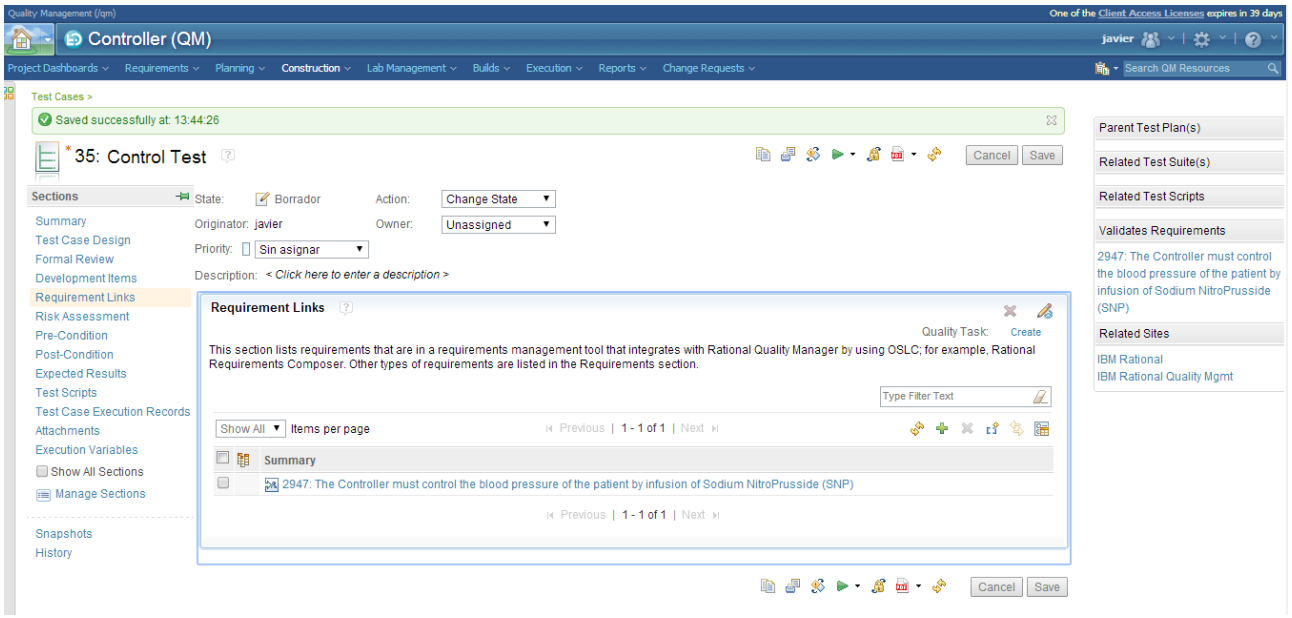

Figure 4-3: Link of the Test Case to the requirement in the QM

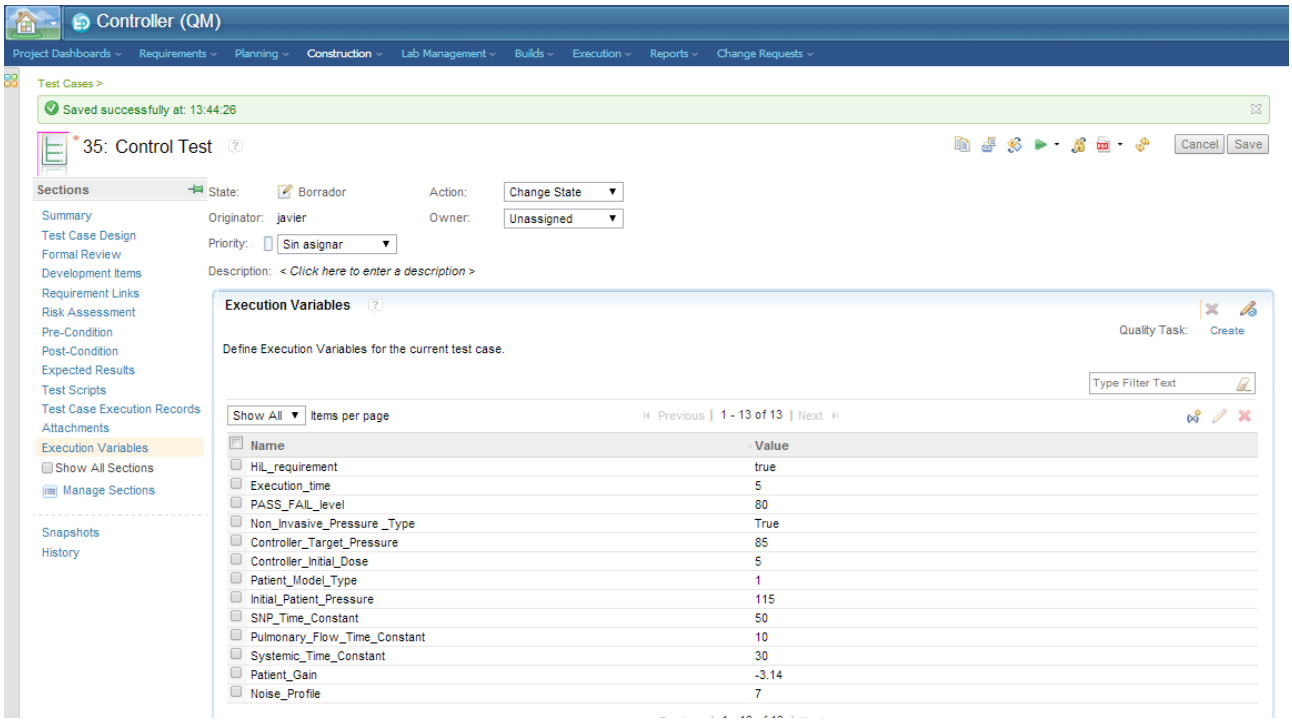

Figure 4-4: List of variable fields in the Test Case definition

4.6.1.3 Scenarios

Scenario Id.	Scenario	Artifacts used	Tools
SC01	Executes a manual test using a HiL as a component in the validation setup	Test definition, Test result	QM, HiL-B4.06
SC02	Executes a set of automatic tests using a HiL as a component in the validation setup	Test list, Test definition, Test result	QM, HiL-B4.06
SC03	Executes a complete manual test using a HiL in the validation setup		HiL-B4.06

4.6.1.3.1 Scenario 1 - SC1

Scenario description: Executes a manual test using a HiL as a component in the validation setup
Related user story:
Related Engineering Method: UC406_Validation_Plan_Execution_003
Related tool chain: IBM QM, HiL-B4.06
Interoperability requirements: Get and set information from/to IBM QM
Setup:
The test definition is stored in the IBM QM.
The tool is accessible via OSLC.
All the devices required (controller, infusion pump, blood pressure simulator, HiL-B4.06) are available and connected.

Step 1:

The HiL-B4.06 gets the list of available tests from the Test Plan in the QM .

A test is selected and the configuration parameters are received.

The configuration parameters required are:

- Pass/Fail Level
- Patient model type.
- Initial patient pressure
- SNP time constant
- Pulmonary flow time constant
- Systemic time constant
- Patient gain
- Noise profile

Step 2:

The HiL-B4.06 gets configured from the parameters received and waits for the start command though the user interface.

Step 3:

The controller is set manually with parameters required to start the control. These parameters are obtained from the test case selected in the QM and are:

- Target value of pressure
- Initial dose of infused drug

Step 4:

The controller and HiL-B4.06 are started simultaneously. When the start command is received the HiL-B4.06 start working (model is initiated) and records the required information. Patient pressure and SNP infusion rate are stored periodically. Information about relevant events (noise or others) is stored asynchronously

Step 5:

When the HiL-B4.06 reaches the execution time set in the Test Case the HiL-B4.06 stops the model execution.

Step 6:

The result condition (PASS/FAIL) is calculated based on the information recorded.

Step 7:

The result is updated and recorded in the QM. Also the information recorded and used to calculate the result are updated and recorded in the QM. Other relevant data are automatically stored in the QM (date and time of test, test engineer, etc.)

Results:

As a result of the scenario, the result field in the Test Case definition is updated.

4.6.1.3.2 Scenario 2 – SC2

Scenario description: Executes a set of automatic tests using a HiL as a component in the validation setup

Related user story:

Related Engineering Method: UC406_Validation_Plan_Execution_003

Related tool chain: IBM QM, HiL-B4.06

Interoperability requirements: Get and set information from/to IBM QM

Setup:

The Test Plan including the definition of all tests is stored in the IBM QM.

The tool is accessible via OSLC.

All the devices required (controller, infusion pump, blood pressure meter, HiL-B4.06) are available and connected.

Step 1:

The HiL-B4.06 gets the Plan Test from the QM. A set of tests is selected automatically based on the fact that the test requires the use of the HiL as a component (HiL_required field in the Test Case definition).

Step 2:

A test is selected from the test list. If the list is empty go to Step 10.

Step 3:

The test configuration parameters are received. The parameters required are:

- Pass/Fail Level
- Patient model type.
- Initial patient pressure
- SNP time constant
- Pulmonary flow time constant
- Systemic time constant
- Patient gain
- Noise profile

Also the parameters required to start the controller are received. These parameters are:

- Target value of pressure
- Initial dose of infused drug

Step 4:

The HiL-B4.06 gets configured from the parameters received. The HiL-B4.06 sends to the controller the two parameters required to start the control session. This information is sent to the controller through a serial channel. When the HiL-B4.06 starts it also send start command to the controller through the same channel.

Step 5:

The HiL-B4.06 start working (model is initiated) and records the required information. Patient pressure and SNP infusion rate are stored periodically. Information about relevant events (noise or others) is stored asynchronously.

Step 6:

When the HiL-B4.06 reaches the execution time set in the Test Case the HiL-B4.06 stops the model execution.

Step 7:

The result condition (PASS/FAIL) is calculated based on the information recorded.

Step 8:

The result is updated and recorded in the QM. Also the information recorded and used to calculate the result are updated and recorded in the QM. Other relevant data are automatically stored in the QM (date and time of test, test engineer, etc.)

Step 9:

The test is removed from the list and go to Step 2.

Step 10:

Test execution has finished.

Results:

As a result of the scenario, the result field in the Test Case definition of all the tests executed is updated.

4.6.1.3.3 Scenario 3 – SC3

Scenario description: Executes a complete manual test using a HiL in the validation setup

Related user story:

Related Engineering Method: UC406_Validation_Plan_Execution_003

Related tool chain: HiL-B4.06

Interoperability requirements: None

Setup:

All the devices required (controller, infusion pump, blood pressure meter, HiL-B4.06) are available and connected.

Step 1:

The HiL is manually configured using the user interface. The parameters required are:

- Pass/Fail Level
- Patient model type.
- Initial patient pressure
- SNP time constant
- Pulmonary flow time constant
- Systemic time constant
- Patient gain
- Noise profile

Step 2:

The controller is set manually with parameters required to start the control. These parameters are obtained from the test case selected in the QM and are:

- Target value of pressure
- Initial dose of infused drug

Step 3:

The controller is started manually. The HiL-B4.06 start working when the start command is manually issued using the user interface (model is initiated) and records the required information. Patient pressure and SNP infusion rate are stored periodically. Information about relevant events (noise or others) is stored asynchronously.

Step 3:

The end of the test is manually issued using the user interface.

Step 4:

The result condition (PASS/FAIL) is calculated based on the information recorded.

Step 5:

The result is shown in the user interface

Results:

As a result of the scenario the result of the test is known.

5 Terms, Abbreviations and Definitions

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

COTS	Custom off the shelve
ICU	Intensive Care Unit
IOS	Interoperability Specification
QM	Quality Manager
TBD	To Be Defined

Table 5: Terms, Abbreviations and Definitions

6 References

Please add citations in this section.

[Author, Year]	Authors; *Title*; Publication data (document reference)

7 Annex I: Detailed Descriptions of the Engineering Methods

These are captured by the Excel templates. The Excel files will be inserted here, when this document is in the final version before it is submitted to the ARTEMIS JU

7.1 Engineering Method: Model Based Analysis

Engineering Method: UC406_Model_Based_Analysis_001					
Purpose:					
Comments:					
Pre-Condition		**Engineering Activities (made of steps)**		**Post-Condition**	
Notes:		Notes:		Notes:	
Artefacts Required as inputs of the Activities		**Artefacts used internally within the Activities (optional)**		**Artefacts Provided as outputs of the Activities**	
Name		Name		Name	
Generic Type: (Tool or language independend type)		Type:		Generic Type: (Tool or language independend type)	
Required Properties: (Information required in interactions between steps)		Properties:		Provided Properties: (Information provided in interactions between steps)	
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	
Name		Name		Name	
Generic Type: (Tool or language independend type)		Type:		Generic Type: (Tool or language independend type)	
Required Properties: (Information required in interactions between steps)		Properties:		Provided Properties: (Information provided in interactions between steps)	
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	
Name		Name		Name	
Generic Type: (Tool or language independend type)		Type:		Generic Type: (Tool or language independend type)	
Required Properties: (Information required in interactions between steps)		Properties:		Provided Properties: (Information provided in interactions between steps)	
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	

7.2 Engineering Method: Validation Plan Definition

Engineering Method: UC406_Validation_Plan_Definition_002					
Purpose: Definition of the validation plan and the requirements for the tools needed for the execution of such a plan					
Comments:					
Pre-Condition		**Engineering Activities (made of steps)**		**Post-Condition**	
System Requirements are available in a Requirement Management Tool (RMT)		1. Creation of the test cases based on the requirements and incorporate it to the Test Management Tool (TMT) 2. Creation of the specification for the Test Execution Controller (TEC) to execute test cases automatically 3. Creation of the specification of Simulators (HiL simulator) required to execute test plan. It includes: 3.1 Specification of Models required 3.2 Specification of Hardware platfform		1. Test cases defined and stored in TMT 2. TEC specification 3. Simulators specification	
Notes:		Notes:		Notes:	
Artefacts Required as inputs of the Activities		**Artefacts used internally within the Activities (optional)**		**Artefacts Provided as outputs of the Activities**	
Name	Requirements defined in natural languaje	Name		Name:	Test Cases
Generic Type:		Type:		Generic Type:	
Required Properties:	Every requirement must include: - Identifier - Description - Origin of the requirement	Properties:			Validation plan consists of a set of test cases. Every test case must include: - Definition - Pass/Fail Condition - Set-up configuration - Test result (no value until the test is executed)
Description & Interoperability Additional constraints:		Description & Interoperability Additional constrains:		Description & Interoperability Additional constraints:	
Name:		Name:		Name:	TEC Specification
Generic Type:		Generic Type:		Generic Type:	
Required Properties:		Required properties:			Specification of the device that controls automatic test execution by setting simulators and external peripherals according to the Test Definition.
Description & Interoperability Additional constraints:		Description & Interoperability Additional constrains:		Description & Interoperability Additional constraints:	
Name:		Name:		Name:	Simulator Specifications
Generic Type:		Generic Type:		Generic Type:	
Required Properties:		Required properties:			Specification of the simulators required for the test execution. Includes hardware and model definition.
Description & Interoperability Additional constraints:		Description & Interoperability Additional constrains:		Description & Interoperability Additional constraints:	

7.3 Engineering Method: Validation Plan Execution

Engineering Method: UC406_Validation_Plan_Execution_003					
Purpose: Execute the Validation Plan					
Comments: Includes the communication with the HiL used for the validation execution					
Pre-Condition		**Engineering Activities (made of steps)**		**Post-Condition**	
1. Validation Plan in a Test Management Tool (TMT) 2. System under validation whith all its components (hardware and software) 3. Simulator(s) required for the execution of test cases 4. Test Execution Controller (TEC) for automatic test case executions		1. Install the setup required for Validation (system under test and all peripherals and measurement devices required) 2. TEC gets the test case set-up for from the Validation Plan 3. The validation consist in a set of test cases to be executed manually or automatically 4. For each automatic test, TEC configures the devices according to the Test Definition (that includes HiL simulators) and executes it. 5. For each manual test, the result is manually updated in the TMT 6. TEC records the result of the validation activities and send the results to the TMT 7. A final report is generated		1. System validated with a list of known defects 2. Test Execution log 3. Validation Execution Result Report	
Notes:		Notes:		Notes:	
Artefacts Required as inputs of the Activities		**Artefacts used internally within the Activities (optional)**		**Artefacts Provided as outputs of the Activities**	
Name:	Test Definitions	Name:		Name:	Test Plan
Generic Type:		Type:		Type:	
Required Properties:	Validation plan is made of a set of tests. Every test must include: - Definition - Pass/Fail conditions - Set-up configuration - Test result (no value until the test is executed)	Properties:	TBD	Propierties:	Test case result field must be updated with the result of the test execution for each executed tests from the plan.
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	
Name:	Device set-up				
Generic Type:					
Required Properties:	Information about how to configure a specific device (simulator or instrument) required to execute a test.				
Description & Interoperability Additional Constraints:					
Name:					
Generic Type:					
Required Properties:					
Description & Interoperability Additional Constraints:					

7.4 Requirement Traceability

Engineering Method: UC406_Requirement_Traceability_004					
Purpose:					
Comments:					
Pre-Condition		**Engineering Activities (made of steps)**		**Post-Condition**	
Notes:		Notes:		Notes:	
Artefacts Required as inputs of the Activities		**Artefacts used internally within the Activities (optional)**		**Artefacts Provided as outputs of the Activities**	
Name		Name		Name	
Generic Type: (Tool or language independend type)		Type:		Generic Type: (Tool or language independend type)	
Required Properties: (Information required in interactions between steps)		Properties:		Provided Properties: (Information provided in interactions between steps)	
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	
Name		Name		Name	
Generic Type: (Tool or language independend type)		Type:		Generic Type: (Tool or language independend type)	
Required Properties: (Information required in interactions between steps)		Properties:		Provided Properties: (Information provided in interactions between steps)	
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	
Name		Name		Name	
Generic Type: (Tool or language independend type)		Type:		Generic Type: (Tool or language independend type)	
Required Properties: (Information required in interactions between steps)		Properties:		Provided Properties: (Information provided in interactions between steps)	
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	

7.5 Impact Analysis

Engineering Method: UC406_Impact_Analysis_005					
Purpose: Impact Analysis					
Comments: Process to accept or deny proposed changes. It also updates the requirements accordingly whether if the change is accepted or not (for ensuring requirements traceability).					
Pre-Condition		**Engineering Activities (made of steps)**		**Post-Condition**	
1. Change proposal due: - Client request (internal or external) - Norm change		1. Analyse requirements affected by the change - Import requirements list (from a Requirement Management tool) - Identify requirements affected by the change 2. Impact analysis (cost, time and risk) 3. Cost of not applying the change (if necessary) 4. Change evaluation (accept or reject) - Create a report (report attached to the change request) - Change Committee decides based on the report (there are several change committees depending on the impact level one or another takes the decision). - If a change request proposed by a client is rejected, discuss with him for seeing alternatives. May be this can lead to a new change request. 5. Update requirements affected (if change is accepted) 6. Notify Change Requirement Downwards in the Development Process (Architecture, Validation, etc) - Using IOS notify used tools about this change requirement (Pull or Push approach) - This will probably trigger the start of a new development cycle and the generation of a new product version (including one or several change requests)		1. Change accepted or denied 2. If accepted: - Requirements updated in the Requirements Management Tool - And a new development process started - Architecture and Validation must be updated according the new requirements. 3. If rejected: - Changes stored. No changes.	
Notes:		Notes: Each requirement has a field with the identifier of the last change request that affected it. Each time this field is updated a new entry in the log history is stored for tracking the previous changes that have modified this requirement.			
Artefacts Required as inputs of the Activities		**Artefacts used internally within the Activities (optional)**		**Artefacts Provided as outputs of the Activities**	
Name:	Change proposal	Name:		Name:	Change proposal - accepted or denied
Generic Type:		Type:		Type:	
Required Properties:	- Change Id - Change Reason - Change Requester - Date - Change Status (proposed) - Change Impact Analysis (empty at this stage: requirements affected, cost, time and cost of not performing it) - Change Decision Description	Properties:		Propierties:	- Change Id - Change Reason - Change Requester - Date - Change Status (accepted or denied) - Change Impact Analysis - Change Decision Description
Description & Interoperability Additional Constraints:		Description:		Description & Interoperability Additional Constraints:	
Name:	Requirements			Name:	Requirement updated
Generic Type:				Generic Type:	
Required Properties:	Every requirement must include: - Identifier - Description - Origin of the requirement			Propierties:	Every requirement must include: - Identifier - Description - Origin of the requirement
Description & Interoperability Additional Constraints:				Description & Interoperability Additional Constraints:	

8 Annex II: Technology Base Line & Progress Beyond

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

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