ICT-2011-288048

EURECA

Enabling information re-Use by linking clinical Research and CAre

IP
Contract Nr: 288048

Deliverable 8.1: Evaluation and validation procedures for the EURECA environment

Due date of deliverable: (31-12-2012)
Actual submission date: (12-03-2013)

Start date of Project: 01 February 2012 Duration: 42 months

Responsible WP8 FORTH

Revision: <outline, draft, **proposed**, accepted>

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination level		
PU	Public	x
PP	Restricted to other programme participants (including the Commission Service	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (excluding the Commission Services)	

0 DOCUMENT INFO

0.1 Author

Author	Company	E-mail
Manolis Tsiknakis	FORTH	tsiknaki@ics.forth.gr
Haridimos Kondylakis	FORTH	kondylak@ics.forth.gr
Lefteris Koumakis	FORTH	koumakis@ics.forth.gr
Manolis Daskalakis	FORTH	emdask@ics.forth.gr
Pascal Coorevits	EUROREC	pascal.coorevits@exchange.eurorec.org

0.2 Documents history

Document version #	Date	Change
V0.1	01-06-2012	Starting version, template
V0.2	01-07-2012	Definition of ToC
V0.3	30-10-2012	First complete draft
V0.4	30-11-2012	Integrated version (send to WP members)
V0.5	10-12-2012	Updated version (send PCP)
V0.6	20-12-2012	Updated version (send to project internal reviewers)
Sign off	31-01-2012	Signed off version (for approval to PMT members)
V1.0	01-03-2013	Approved Version to be submitted to EU

0.3 Document data

Keywords	
Editor Address data	Name: Lefteris Koumakis Partner: FORTH Address: N. Plastira 100 Vassilika Vouton Heraklion Phone: +30 2810 391424 Fax: +30 2810 391448 E-mail: koumakis@ics.forth.gr
Delivery date	13-03-2013

0.4 Distribution list

Date	Issue	E-mailer
13-03-2013	1.0	Benoit.ABELOOS@ec.europa.eu
		INFSO-ICT-288048@ec.europa.eu

Executive Summary

This document establishes the evaluation process, defines the evaluation criteria and makes a preliminary identification of the products to be evaluated. The evaluation process is based on the ISO/IEC 25000 standard, so the relevant evaluation criteria are used. However, in general, we expect that evaluation criteria will be continuously adapted to the current state of development of the environment, considering the end-user scenarios and clinical pilots as general guideline.

Moreover, this deliverable presents an overview of the validation methodology that we intent to use within the EURECA project.
There are many approaches that can be combined to conduct validation activities and tests, depending on the constraints. Different approaches can be combined to the requirements for different types of service, service model, risk profile, skill levels, test objectives and levels of testing. Examples include:

- Modelling and measuring – suitable for testing the service model and Service Operations plan.
- Risk-based approach that focuses on areas of greatest risk, e.g. business critical services, risks identified in change impact analysis.
- Simulation
- Scenario testing and
- Live pilot.

The decision of the EURECA consortium is that live pilot validation activities will be executed, using a range of test cases with known results within and without the technological platform established within EURECA. The specific details of the test cases will be defined in a subsequent deliverable of WP8.

Table of Contents

1 Introduction

Evaluation is the systematic determination of the extent to which an entity meets its specified criteria. The evaluation of software product quality is vital to both the acquisition and development of software. The relative importance of the various characteristics of software quality depends on the intended usage or objectives of the system of which the software is a part; software products need to be evaluated to decide whether relevant quality characteristics meet the requirements of the system.

In general terms, the quality expectations for software systems are twofold:

- the software must do the right things: software systems must do what they are supposed to do (end-user perspective)
- the software must do the things right: software systems must perform the tasks correctly (developer perspective)

These two aspects define two of the main components of the software quality assurance system (SQAS): the validation (does the software do the right things?) and verification (does the software do the things right?).

Accordingly, SQAS aims at ensuring a high quality of the software product through the related validation and verification activities. These activities must be carried out by the people and the organizations responsible for developing and supporting the system in an overall engineering process that includes:

- Quality planning
- Execution of selected quality assurance activities
- Measurement and analysis to demonstrate software quality to all parties involved.

Unfortunately, as the complexity and code size of the software increase, the risks of having a failure increase as well, and there is no effective general solution to the size, complexity, quality and other software engineering problems. However, by following standardized software development practices and by addressing the quality issues during the whole life cycle of the software, the likelihood of such defects and the cost incurred by them (both to users and to producers) may be greatly reduced.

1.1 Purpose of the evaluation

The purpose of this document is to propose a unified approach for ensuring the quality of the software products produced within the EURECA project, in accordance with the guidelines established in Deliverable (D1.1). So, in this document the procedures for the evaluation and validation activities will be established and qualitative measures of the benefits of the project as a whole will be developed.

The implementation of this approach is adapted from various sources and mainly from the ISO/IEC 25000 series. Due to the high complexity of the software to be produced/ integrated within the EURECA project, this document does not attempt to cover all possible aspects of quality monitoring/ensuring for every module, but rather provide a template that should be adapted at the level of each module.

1.2 Quality requirements

The evaluation of the EURECA modules, tools and components should clearly be viewed as an iterative process. *Scenarios* and *Quality Assurance* procedures will evolve as new components get integrated in the environment or as some others are removed if considered useless. This evolution will be depicted in the deliverables following D8.1 within the Q&A, Evaluation and Validation work package.

1.3 Document Structure

The structure of this document is the following: Section 2 presents an overview of the evaluation process, and presents the products to be evaluated. Those products in EURECA will be individual components and clinical services. A preliminary list of these components and clinical services is presented in this section as well. Then, Section 3 presents the evaluation model and defines the decision criteria adopted. Section 4 presents the validation procedures and then and Section 5 concludes this deliverable and links this deliverable to the other deliverables of WP8.

2 Establish Evaluation Requirements

For quality assurance, within the EURECA project, norms defined from the International Organization for Standardization[1] (ISO) will be used. More specifically the Software Product Quality Requirements and evaluation (SQUARE) will be used as a reference model, shown also in Figure 1. It describes the general processes and details the activities and tasks providing their purposes, inputs, outcomes and complementary information that can be used to guide a software product quality evaluation. It is actually the new version of the ISO/IEC 14598.

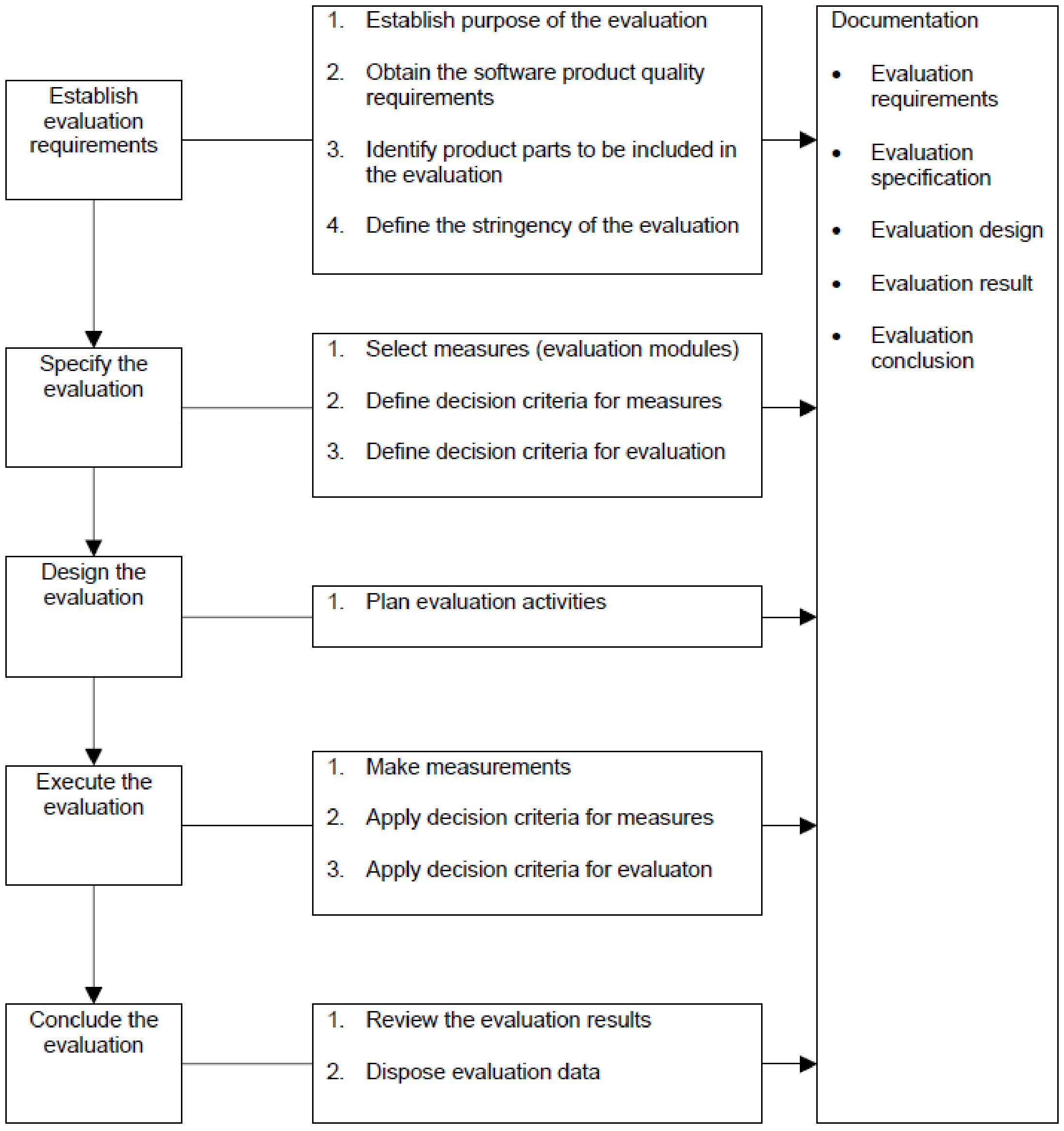
Establish evaluation requirements
Specify the evaluation
Design the evaluation
Execute the evaluation
Conclude the evaluation
1. Establish purpose of the evaluation
2. Obtain the software product quality requirements
3. Identify product parts to be included in the evaluation
4. Define the stringency of the evaluation
1. Select measures (evaluation modules)
2. Define decision criteria for measures
3. Define decision criteria for evaluation
1. Plan evaluation activities
1. Make measurements
2. Apply decision criteria for measures
3. Apply decision criteria for evaluaton
1. Review the evaluation results
2. Dispose evaluation data
Documentation
Evaluation requirements
Evaluation specification
Evaluation design
Evaluation result
Evaluation conclusion

Figure 1: Software product Quality Evaluation Process reference model adapted from ISO/IEC 25040

[1] http://www.iso.org

The main blocks of the evaluation process are “Establish evaluation requirements”, “Specify the Evaluation”, “Design the evaluation”, “Execute the evaluation” and “Conclude the evaluation”. In this document the first two blocks will be defined and analyzed and the rest of them will be analyzed in the next deliverables of WP8.

2.1 Products to be evaluated

This Section describes the identified “products” of the EURECA project to be evaluated. These “products” are individual technological components and cross-cutting clinical services. However since the deliverable which is related to the architecture (D2.2) is to be delivered at month 12, the list of the identified components and clinical services to be evaluated is not final. Moreover we expect that as the projects progresses through time, requirements might change. Nevertheless we present here an initial list of the components and clinical services to be evaluated.

2.1.1 Technological Components

- **EURECA local DWH:** Each organization will have locally installed a EURECA-compatible data warehouse where relevant data will be stored. The data warehouse will offer services for accessing and storing data.
- **Literature DBs**: Those DBs will offer access to the literature.
- **Personal Medical Information Recommender**: This is a tool that will recommend to the patient relevant information. Information will come from literature searching.
- **Update guidelines tool:** This tool will allow one to adjust a clinical guideline based on evidences from literature.
- **Broad Consent tool:** This tool will allow patients to easily consent to broad use of their data.
- **Hypothesis Generation tool:** This tool will allow one to generate hypotheses from existing patient data.
- **Protocol feasibility tool:** This tool will allow Clinical Trial Managers to design or assess the feasibility of a new trial using existing patient data.
- **Microbiology SAE:** This tool will allow one to find easily serious adverse events using microbiology data.
- **Outcome Prediction:** This tool will allow predicting the outcome of an individual patient.
- **Diagnostic Sarcoma Classifier tool:** This tool will be used to diagnose different types of sarcoma.
- **Open DBs:** Those DBs will offer publicly available data sets.

- **Anonymization tool:** Anonymization Tool is a service-based de-identification solution. It will support anonymization of different types of data (XML, DICOM, Text, Database, etc.) in a generic and extendable way.

- **HIS/EHR/PHR systems:** These systems are used in order to capture patient records and their relevant data such as medications, visits, surgeries, laboratory data etc.

- **Patient Identity Management Service:** Patients selected for trial screening are managed in the EURECA platform in the patient identity management service. It is responsible for the registration, consultation and editing of patient meta-data relevant for the EURECA patient screening (real patient data is part of the EHR DW)

- **Trial Management Service:** This service is responsible for providing trial registering, querying and editing functionality to a site. It offers registering services that enables a trial administrator to generate general trial information, add eligibility criteria to a trial, define different trial arms in a trial, etc. All this trial information is stored in a trial meta-data repository of the site. Another important service is that the information stored in the trial repository, for example the list of trials, can be easily accessed by other services of the site.

- **Criteria Matcher:** An eligibility criterion is matched with the information of a selected patient in the criteria matching service. It provides an interface that enables other EURECA services (in our case the screening service) to send matching requests. The criteria matcher will query the requested information of the patient by sending the query that is included in the eligibility criterion to the CIM based query service of the different available data-warehouses. The eligibility criterion itself is retrieved from the trial management service. The outcome of the matching is sent back to the requesting service.

- **Common Information Model-Based Data Access:** This service provides functionality to query the datasets of the EHR and other data warehouses available on the site through the semantic layer. It abstracts the underlying data sources for the upper EURECA services (in our case, the screening service) and presents data to applications according to a single integrated data model. More information about this service can be found in the semantic layer view section.

- **Freetext Query Service:** This service is responsible for freetext querying of the different available datasources (e.g. the EHR data warehouse) in the EURECA platform. It offers freetext searching functionality in order to query structured and unstructured data.

- **Cancer Registry Reporting tool:** This tool will allow one to report patients to the cancer registry by re-using data already collected.

- **Automatic SAEs/SUSARs tool:** This tool will automatically file a SAE/SUSAR report by re-using already collected data.

2.1.2 Clinical Services

Bellow we describe the clinical services that will be implemented in the EURECA platform. Note that these general services contain many sub-services that we expect to be selected and described in detail as the project progresses in time.

- **Information:** This service will be used to achieve several goals.
    - First personal medical information can be recommended to a doctor or a patient. A better characterization of a patient according to his/her risk factors will help to predict the outcome of the disease for him/her. This can also be seen as a simulation of the response to different treatments and can be done by selecting patients with the same characteristics from the database and show which treatment results in which outcome. A search for further risk criteria will help to distinguish these patients into more different prognostic groups, to find for an individual patient the optimal treatment.
    - Moreover since many questions asked by patients are repeated into consultation a mechanism for automatic generating answers to those questions will be implemented.

- **Investigation:** This service is composed of a series of other sub-services focusing either on Clinical Guidelines Investigation or in Protocol & Research Investigation:
    - Clinical Guidelines Investigation focuses on the update of already established Guidelines based on data mining from CT/HIS databases, literature and trial databases. In this task classifiers can help that will be trained, validated and updated.
    - Concerning Protocol & Research Investigation focuses on patient and trial management"
        - First a mechanism to allow patients share (or not) their data should be implemented. So, a service allowing informed consent to be signed and updated should be available.
        - Moreover, before starting a new clinical trial a new research question is needed. Such a question is of utmost importance and is part of hypothesis generation. Analysing all available data from previous trials, guidelines, literature and others, can support this process. It can also help to find biomarkers that are relevant for the disease suggesting their use in the trial for evaluation or validation purposes. The hypothesis generation assistance can support the design of new trials.
        - When a clinical trial is being designed a protocol feasibility service will identify if a new clinical trial is feasible to start according to the estimation of recruitment potential. It will be based either on EHR/PHR/HIS data or other fata sources such as public data, population information other protocols or literature.

- **Selection & Recruitment:** This service focuses on the choice of optimal treatment for a patient and the selection of the appropriate trial to be enrolled.
    - The early knowledge about infectious agents and their resistance profile for patients in chemotherapy is really important for the optimal treatment

 - of a patient. Common Toxicity Criteria can be specified in order to detect SAE events automatically.
 - Moreover, we want to learn and validate outcome prediction models from routine patient care data. We need to have access to large amounts (10.000+) of patient's data preferentially with clinical, imaging, biology information. This scenario can be integrated in the scenario 'Personal medical information recommender'.
 - Concerning patient recruitment the purpose of this service is to identify eligible patients for clinical trials, or vice versa. The goal is to find the optimal trial that fits the needs of the patient the best.

- **Reporting:** The purpose of this service is to detect and report
  - Information about specific tumour from the local cancer registry including all patient information.
  - Information stored in local IT systems about patients in order to avoid double data entry for the clinical trial management systems.
  - Episodes of febrile neutropenia by extracting some specific symptoms and clinical relevant characteristics from EHR on a given period of time for retrospective study.
  - SAE and SUSARs based on a database of pharmacogenomics

- **Long-Term Follow-up:** This service will allow a trial chairman to define follow-up eCRFs. Those eCRFs can be filled either manually using the Clinical Trial Management System or automatically by querying relevant data from patient PHR. Moreover, eCRFs will be possible to be pre-filled with information from national registries.

- **Economic Analysis:** By joining data from EHR, clinical trials, literature and open databases economic aspects of different procedures (diagnostic and/or therapeutic) can be analysed in respect to outcome and quality of life in an individual patient. This will include data about days to stay in the hospital, expected side effects, costs of diagnostics and therapeutics, etc.

3 Evaluation Modules

The goal of the evaluation is to ensure that the software produced in each technical WP is compliant with the end-user specifications.

The evaluation of software product quality is vital to both the acquisition and development of software. The relative importance of the various characteristics of software quality depends on the intended usage or objectives of the system.

Evaluation modules contain the specification of the quality model (i.e. characteristics, sub-characteristics and corresponding internal, external or quality in use measures), the associated data and information about the planned application of the model and the information about its actual application. Appropriate evaluation modules will be selected for the EURECA components evaluation based on the Software product Quality Requirements and Evaluation (SQuaRE) – Evaluation reference model and guide (SQuaRE).

3.1 Decision criteria for measures

ISO and the International Electrotechnical Commission (IEC) form the specialized system for worldwide standardization. The ISO SQuaRE, will be used as reference model. Joint Technical Committee ISO/IEC JTC 1, Information technology, Subcommittee SC 7, Software and systems engineering, prepared ISO/IEC 25010. ISO/IEC 25010 is a part of the SQuaRE series of International Standards, which consists of the following divisions:

- Quality Management Division ISO/IEC (2500n)
- Quality Model Division ISO/IEC (25010)
- Quality Measurement Division ISO/IEC (25020)
- Quality Requirements Division ISO/IEC (25030)
- Quality Evaluation Division ISO/IEC (25040)
- SQuaRE Extension Division ISO/IEC 25050 – ISO/IEC 25099 (to appear)

This first edition of ISO/IEC (25010) cancels and replaces ISO/IEC 9126-1:2001, which has been technically revised.

ISO/IEC 9126:1991 was replaced by two related multipart standards: ISO/IEC 9126, *Software engineering — Product quality* and ISO/IEC 14598, *Software engineering — Product evaluation*. This International Standard revises ISO/IEC 9126-1:2001, and incorporates the same software quality characteristics with some amendments.

- The scope of the quality models has been extended to include computer systems, and quality in use from a system perspective.
- Context coverage has been added as a quality in use characteristic, with sub-characteristics *context completeness* and *flexibility*.
- *Security* has been added as a characteristic, rather than a sub-characteristic of functionality, with sub-characteristics *confidentiality*, *integrity*, *non-repudiation*, *accountability* and *authenticity*.
- *Compatibility* (including *interoperability* and *co-existence)* has been added as a characteristic.
- The following sub-characteristics have been added: *functional completeness, capacity, user error protection, accessibility, availability, modularity* and *reusability*.

- The compliance sub-characteristics have been removed, as compliance with laws and regulations is part of overall system requirements, rather than specifically part of quality.
- The internal and external quality models have been combined as the product quality model.
- When appropriate, generic definitions have been adopted, rather than using software-specific definitions.
- Several characteristics and sub-characteristics have been given more accurate names.

This International Standard defines:

- A product quality model composed of eight characteristics (which are further subdivided into sub-characteristics) that relate to static properties of software and dynamic properties of the computer system. The model is applicable to both computer systems and software products. Section 3.2.1 describes in detail the product quality model.
- A quality in use model composed of five characteristics (some of which are further subdivided into sub-characteristics) that relate to the outcome of interaction when a product is used in a particular context of use. This system model is applicable to the complete human-computer system, including both computer systems in use and software products in use. Section 3.2.2 describes in detail the quality in use model.

3.2 Quality model structure

A quality model is a set of requirements, entities and relationships that must be fulfilled to assess good quality. The model should be structured in three main levels:

- Characteristic
- Sub-characteristic
- Attribute

We can refer to two models of quality:

- the internal and external quality
- the quality in use

The product quality model in categorizes system/software product quality properties into eight characteristics: functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability and portability. Each characteristic is composed of a set of related sub-characteristics (Figure 2Figure 2: Software product quality categories and characteristics (source ISO/IEC 25040)). The product quality model can be applied to just a software product, or to a computer system that includes software, as most of the sub-characteristics are relevant to both software and systems.

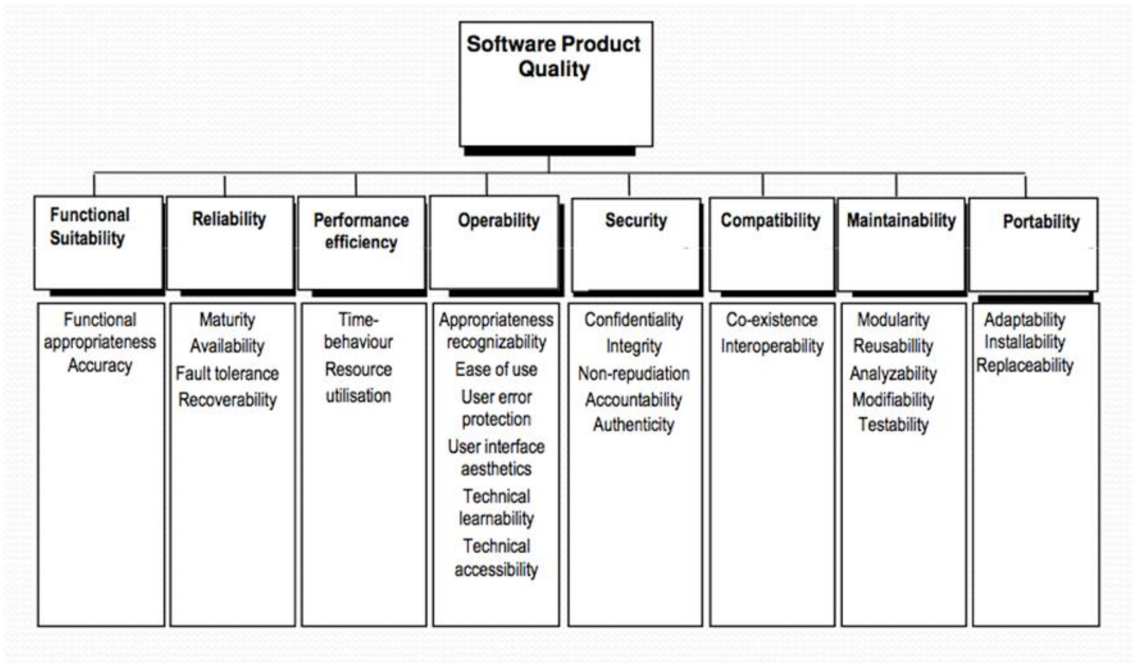

Figure 2: Software product quality categories and characteristics (source ISO/IEC 25040)

3.2.1 Product quality model

The product quality model can be applied to just a software product, or to a computer system that includes software, as most of the sub-characteristics are relevant to both software and systems.

The product quality model categorizes product quality properties into eight characteristics (functional suitability, reliability, performance efficiency, usability, security, compatibility, maintainability and portability). Each characteristic is composed of a set of related sub-characteristics, naming:

Functional suitability
Functional suitability is the degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions

- Functional completeness
  The degree to which the set of functions covers all the specified tasks and user objectives

- Functional correctness
  The degree to which a product or system provides the correct results with the needed degree of precision

- Functional appropriateness
  The degree to which the functions facilitate the accomplishment of specified tasks and objectives
  EXAMPLE: A user is only presented with the necessary steps to complete a task, excluding any unnecessary steps.

Performance efficiency

performance relative to the amount of resources used under stated conditions

- Time behavior
  The degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements

- Resource utilization
  The degree to which the amounts and types of resources used by a product or system when performing its functions meet requirements

- Capacity
  The degree to which the maximum limits of a product or system parameter meet requirements

Compatibility
The degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment

- Co-existence
  The degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product

- Interoperability
  The degree to which two or more systems, products or components can exchange information and use the information that has been exchanged

Usability
The degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use

- Appropriateness recognizability
  The degree to which users can recognize whether a product or system is appropriate for their needs

- Learnability
  The degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use

- Operability
  The degree to which a product or system has attributes that make it easy to operate and control

- User error protection
  The degree to which a system protects users against making errors

- User interface aesthetics
  The degree to which a user interface enables pleasing and satisfying interaction for the user

- Accessibility
  The degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use

Reliability
The degree to which a system, product or component performs specified functions under specified conditions for a specified period of time

- Maturity
  The degree to which a system meets needs for reliability under normal operation

- Availability
  The degree to which a system, product or component is operational and accessible when required for use

- Fault tolerance
  The degree to which a system, product or component operates as intended despite the presence of hardware or software faults

- Recoverability
  The degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system

Security
The degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization

- Confidentiality
  The degree to which a product or system ensures that data are accessible only to those authorized to have access

- Integrity
  The degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data

- Non-repudiation
  The degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later

- Accountability
  The degree to which the actions of an entity can be traced uniquely to the entity

- Authenticity

The degree to which the identity of a subject or resource can be proved to be the one claimed

Maintainability
The degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers

- Modularity
  The degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components

- Reusability
  The degree to which an asset can be used in more than one system, or in building other assets

- Analysability
  The degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified

- Modifiability
  The degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality

- Testability
  The degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met

Portability
The degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another

- Adaptability
  The degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments

- Installability
  The degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment

- Replaceability
  The degree to which a product can be replaced by another specified software product for the same purpose in the same environment
  EXAMPLE The replaceability of a new version of a software product is important to the user when upgrading.

3.2.2 Quality in use model

The quality in use model defines five characteristics related to outcomes of interaction with a system: effectiveness, efficiency, satisfaction, freedom from risk, and context coverage (Figure 3). Each characteristic can be assigned to different activities of stakeholders, for example, the interaction of an operator or the maintenance of a developer.

The quality in use of a system characterizes the impact that the product (system or software product) has on stakeholders. It is determined by the quality of the software, hardware and operating environment, and the characteristics of the users, tasks and social environment. All these factors contribute to the quality in use of the system.

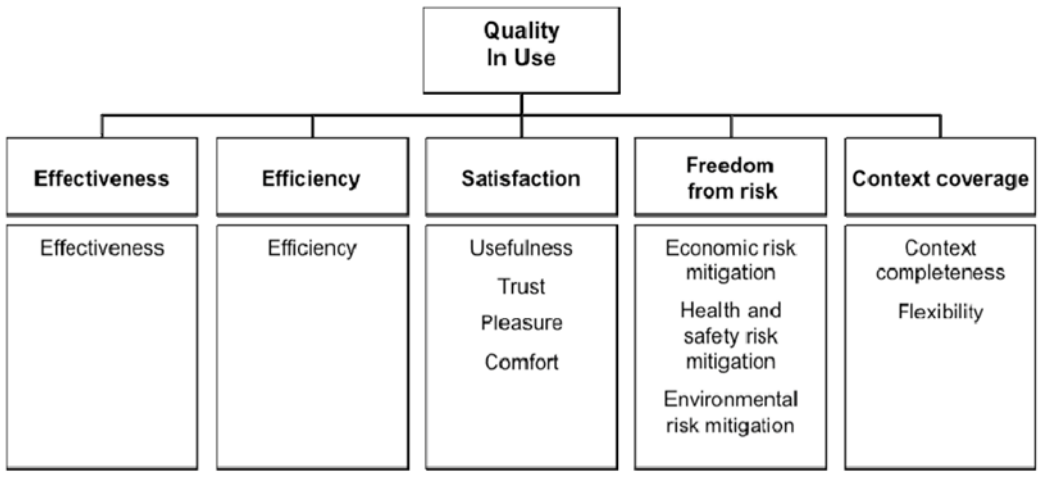

Figure 3: Quality in use model (source ISO/IEC 25010)

Quality in use is the degree to which a product or system can be used by specific users to meet their needs to achieve specific goals with effectiveness, efficiency, freedom from risk and satisfaction in specific contexts of use. The properties of quality in use are categorized into five characteristics: effectiveness, efficiency, satisfaction, freedom from risk and context coverage.

Effectiveness
Accuracy and completeness with which users achieve specified goals

Efficiency
Resources expended in relation to the accuracy and completeness with which users achieve goals

Satisfaction
Degree to which user needs are satisfied when a product or system is used in a specified context of use

- Usefulness
  The degree to which a user is satisfied with their perceived achievement of pragmatic goals, including the results of use and the consequences of use

- <u>Trust</u>
  The degree to which a user or other stakeholder has confidence that a product or system will behave as intended

- <u>Pleasure</u>
  The degree to which a user obtains pleasure from fulfilling their personal needs

- <u>Comfort</u>
  The degree to which the user is satisfied with physical comfort

Freedom from risk
Degree to which a product or system mitigates the potential risk to economic status, human life, health, or the environment

- <u>Economic risk mitigation</u>
  The degree to which a product or system mitigates the potential risk to financial status, efficient operation, commercial property, reputation or other resources in the intended contexts of use

- <u>Health and safety risk mitigation</u>
  The degree to which a product or system mitigates the potential risk to people in the intended contexts of use

- <u>Environmental risk mitigation</u>
  The degree to which a product or system mitigates the potential risk to property or the environment in the intended contexts of use

Context coverage
The degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in both specified contexts of use and in contexts beyond those initially explicitly identified

- <u>Context completeness</u>
  The degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in all the specified contexts of use

- <u>Flexibility</u>
  The degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in contexts beyond those initially specified in the requirements

3.3 Decision criteria for evaluation

The selected software product quality measures shall be applied to the software product and components, according to the evaluation plan, resulting in values on the measurement scales.

None of the quality characteristics discussed above can be measured directly, but must be assessed in terms of objective sub-characteristics. ISO/IEC 25000 series does not prescribe specific quality requirements for software, but instead describes a quality model, which can be applied to any software.

End-user evaluation of the EURECA infrastructure will be conducted through a number of selected scenarios from deliverable (D1.2) covering the anticipated usage of the infrastructure, from administration of the software components to specific clinical trials. For each step in the scenario, the required input data are enumerated and a description of the expected results will be given. The steps listed for the execution of the scenarios respond to criteria which will help objectively rating the degree of success of the modules addressed therein. The end users who will participate at the evaluation phase will fill in an evaluation form for each EURECA component. The evaluation form will cover all the appropriate quality characteristics from the product quality model of the ISO/IEC 25000 series (Figure 2).

At the evaluation phase different type of users, such as physicians, system developers and patients will participate. Having such a diverse target group of evaluators, the evaluation forms must be:

- simple
- accurate
- easy to understand (especially for non IT experts)
- non time consuming
- without loss of functionality/quality

For that reason we have translate the crucial sub-characteristics of software quality measures into simple questions (in natural language). The evaluation form of EURECA will be a list of such questions where the evaluator will answer with a degree of satisfaction with scale 5 (from 1 to 5).

The selected sub-characteristics, for the evaluation form of the EURECA scenarios and components, and its translation into simple statements for the end user can be found in the table below (Those statements will be then rated using a Likert scale by the end users to determine their level of agreement or disagreement on a symmetric agree-disagree scale. The corresponding form generated can be found in the Appendix. However, the idea behind these criteria is that in the next evaluation phase for each one of those the following will be described

- A description of the test procedure.
- A description of the expected results.
- Possible test data to be used.
- External tools for assessment.

Functionality	Completeness	The set of functions covers all the specified tasks and user objectives.
	Correctness	The system provides the correct results with the needed degree of precision.
	Appropriateness	The functions facilitate the accomplishment of specified tasks and objectives.
Efficiency	Time Behaviour	The system responds quickly.

	Resource utilization	The system utilizes resources efficiently.
Compatibility	Co-existence	The system shares resources without loss of its functionality.
	Interoperability	The system shares information/data with other EURECA components?
Usability	Recognisability	The users can recognize easily whether the system is appropriate for their needs.
	Learnability	The users learn to use the system easily.
	Operability	The users use the system without much effort.
	Error protection	The system protects users against making errors.
	UI aesthetics	The user interface enables pleasing and satisfying interaction for the users.
Reliability	Maturity	Most of the faults in the software been eliminated over time.
	Fault tolerance	The software is capable of handling errors.
	Recoverability	The software resumes working & restores lost data after failure.
Security	Authenticity	The system provides identification access wherever is needed.
	Confidentiality	Data are accessible only to authorized users.
	Accountability	The system traces actions uniquely.
	Integrity	The system prevents unauthorized access.
Maintainability	Analysability	Faults can be easily diagnosed.
	Modularity	The system is composed of discrete independent components.
	Reusability	An asset can be used in more than one system, or in building other assets.
	Testability	The software can be tested easily.
Portability	Adaptability	The software can be moved to other environments easily.
	Installability	The software can be installed easily.
	Replaceability	The software can easily replace other software.
Qualit y of use	Effectiveness	The software is accurate and complete for the intended use.

	Efficiency	The software improves the time or reduces resources for the intended goal.
	Satisfaction	The software satisfies the perceived achievements of pragmatic goals.
	Health and safety risk	The software cannot harm people in the intended contexts of use.

Table 1). Those statements will be then rated using a Likert scale[2] by the end users to determine their level of agreement or disagreement on a symmetric agree-disagree scale. The corresponding form generated can be found in the Appendix. However, the idea behind these criteria is that in the next evaluation phase for each one of those the following will be described

- A description of the test procedure.
- A description of the expected results.
- Possible test data to be used.
- External tools for assessment.

Functionality	Completeness	The set of functions covers all the specified tasks and user objectives.
	Correctness	The system provides the correct results with the needed degree of precision.
	Appropriateness	The functions facilitate the accomplishment of specified tasks and objectives.
Efficiency	Time Behaviour	The system responds quickly.
	Resource utilization	The system utilizes resources efficiently.
Compatibility	Co-existence	The system shares resources without loss of its functionality.
	Interoperability	The system shares information/data with other EURECA components?
Usability	Recognisability	The users can recognize easily whether the system is appropriate for their needs.
	Learnability	The users learn to use the system easily.
	Operability	The users use the system without much effort.
	Error protection	The system protects users against making errors.
	UI aesthetics	The user interface enables pleasing and satisfying interaction for the users.

[2] http://en.wikipedia.org/wiki/Likert_scale

Reliability	Maturity	Most of the faults in the software been eliminated over time.
	Fault tolerance	The software is capable of handling errors.
	Recoverability	The software resumes working & restores lost data after failure.
Security	Authenticity	The system provides identification access wherever is needed.
	Confidentiality	Data are accessible only to authorized users.
	Accountability	The system traces actions uniquely.
	Integrity	The system prevents unauthorized access.
Maintainability	Analysability	Faults can be easily diagnosed.
	Modularity	The system is composed of discrete independent components.
	Reusability	An asset can be used in more than one system, or in building other assets.
	Testability	The software can be tested easily.
Portability	Adaptability	The software can be moved to other environments easily.
	Installability	The software can be installed easily.
	Replaceability	The software can easily replace other software.
Quality of use	Effectiveness	The software is accurate and complete for the intended use.
	Efficiency	The software improves the time or reduces resources for the intended goal.
	Satisfaction	The software satisfies the perceived achievements of pragmatic goals.
	Health and safety risk	The software cannot harm people in the intended contexts of use.

Table 1: From software quality characteristics to NL questions

The scenarios for the evaluation will be described in detail in the deliverable (D8.2).

3.4 Rating levels for metrics

To assess quality levels the end user and/or evaluator have already a list of metrics that she/he can measure. A scale must also have been defined; usually the scales can be divided into categories corresponding to different degrees of satisfaction of the requirements like:

- **Minimum level:** none of the relevant quality characteristics should measure below the Minimal level. In case one single characteristic scores below its minimum level, the project failed and the product is unusable.
- **Current level**: when the software product replaces a current situation, a current level is also available. In general, the acceptable level will be equal to or higher than the current level.
- **Acceptable level**: if all characteristics score above the acceptable level, the product has passed the test.
- **Target level**: each quality characteristic should have a challenging target level.
- **Maximum level**: this is a theoretic level and describes the upper limit of what is possible.

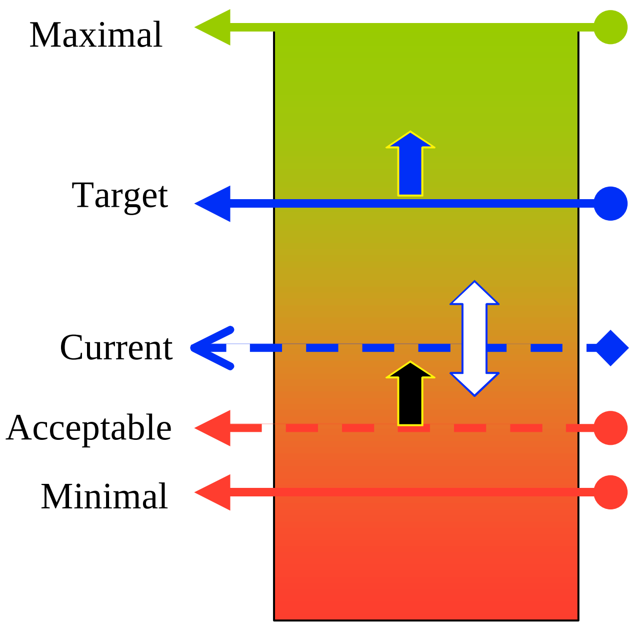

Figure 4: Degrees of satisfaction and interpretations (adapted from ISO/IEC)

The categories should be specified so that both the user and the developer can avoid unnecessary cost and schedule overruns. The relative importance of each quality characteristic determines the minimum, acceptable and target levels for the quality characteristic. The required levels of quality characteristics will then determine how the developers' time will be divided.

When the software product is validated, all tests are carried out and the measured levels are compared with the predefined minimum, acceptable and target level. This will then be the acceptance test.

4 Validation procedures

Trying to answer the question if the software does the right things, an integration of software life cycle management and risk management activities is recommended. Validation of software has been conducted in many segments of the software industry for many years. However a general application of several broad concepts can be used successfully as guidance for software validation. These broad concepts provide an acceptable framework for building a comprehensive approach to software validation.

4.1 General principles and life cycle

It is preferable that guidance on validation should not recommend any specific life cycle model or any specific technique or method. It should recommend that software validation and verification activities must be conducted throughout the entire software life cycle. According to (ISO/IEC12207:2008), software life cycle processes define a common framework, with well-defined terminology, that can be referenced by the software industry and contains processes, activities, and tasks. Software lifecycle applies to the acquisition of systems and software products, to the supply, development, operation, maintenance, and disposal of software products and the software portion of a system, whether performed internally or externally to an organization.

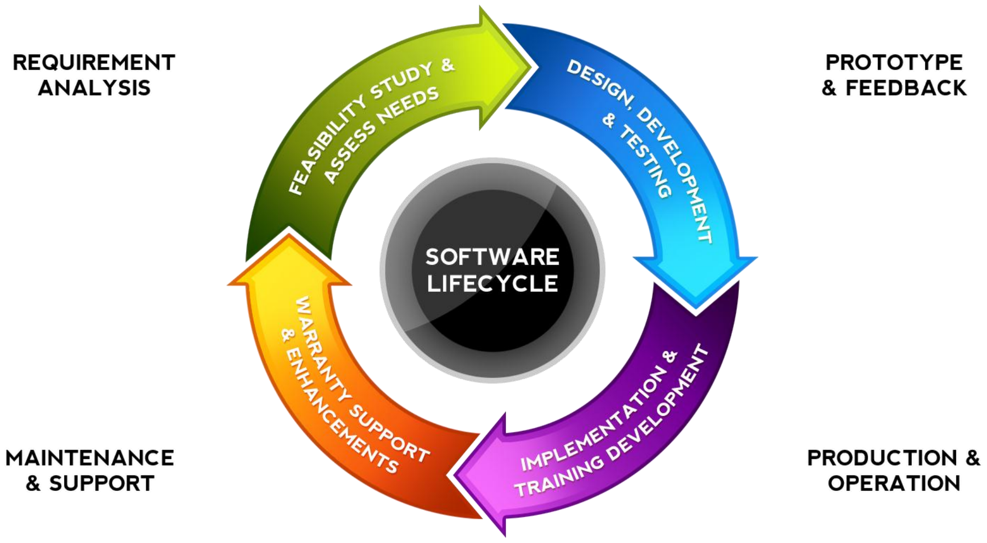
REQUIREMENT ANALYSIS
FEASIBILITY STUDY & ASSESS NEEDS
DESIGN, DEVELOPMENT & TESTING
PROTOTYPE & FEEDBACK
SOFTWARE LIFECYCLE
WARRANTY SUPPORT & ENHANCEMENTS
IMPLEMENTATION & TRAINING DEVELOPMENT
MAINTENANCE & SUPPORT
PRODUCTION & OPERATION

Figure 5: Software life cycle (source http://isoftdev.eu/software/)

Software validation is accomplished through a series of activities and tasks that are planned and executed at various stages of the software development life cycle. Software developers should establish a software life cycle model that is appropriate for their product.

Activities in a typical software life cycle model include the following:

- Quality Planning
- System Requirements Definition
- Detailed Software Requirements Specification
- Software Design Specification

- Construction or Coding
- Testing
- Installation
- Operation and Support
- Maintenance
- Retirement

Verification, testing, and other tasks that support software validation, occur during each of these activities.

4.2 Verification and validation

Software verification and validation are difficult because a developer cannot test forever, and it is hard to know how much evidence is enough. In large measure, software validation is a matter of developing a “level of confidence” that the outcome meets all requirements and user expectations for the software automated functions and features of the system.

According to the Quality System regulation (ISO8402:1994), “verification” and “validation” are treated as separate and distinct terms. On the other hand, many software engineering journal articles and textbooks use the terms "verification" and "validation" interchangeably, or in some cases refer to software "verification, validation, and testing (VV&T)" as if it is a single concept, with no distinction among the three terms. Software validation is a part of the design validation for a production system, but is not separately defined in the Quality System regulation. The implementation of our approach to evaluation is adapted from various sources and mainly from the ISO/IEC 25000 series.

For purposes of guidance, we consider software validation to be “*confirmation by examination and provision of objective evidence that software specifications conform to user needs and intended uses, and that the particular requirements implemented through software can be consistently fulfilled*” (FDA, 1997). In practice, software validation activities may occur both during, as well as at the end of the software development life cycle to ensure that all requirements have been fulfilled.

Software validation is a critical tool used to assure the quality of desired outcome and software automated operations. Software validation can increase the usability and reliability of the system, resulting in decreased failure rates, fewer recalls and corrective actions and less risk to patients and users. Software validation can also reduce long term costs by making it easier and less costly to reliably modify software and revalidate software changes.

The following list summarizes the general principles that should be considered for the validation of a system/software.

- ***Requirements***: A documented software requirements specification provides a baseline for both validation and verification. The software validation process cannot be completed without an established software requirements specification.
- ***Defect Prevention***: Software quality assurance needs to focus on preventing the introduction of defects into the software development process and not on

trying to “test quality into” the software code after it is written. Software testing is very limited in its ability to surface all latent defects in software code.

- ***Time and Effort***: To build a case that the software is validated requires time and effort. Preparation for software validation should begin early, i.e., during design and development planning and design input. The final conclusion that the software is validated should be based on evidence collected from planned efforts conducted throughout the software lifecycle.
- ***Software Life Cycle***: Software validation takes place within the environment of an established software life cycle. The software life cycle contains software engineering tasks and documentation necessary to support the software validation effort. In addition, the software life cycle contains specific verification and validation tasks that are appropriate for the intended use of the software.
- ***Plans***: The software validation process is defined and controlled through the use of a plan. The software validation plan defines “what” is to be accomplished through the software validation effort.
- ***Procedures***: The software validation process is executed through the use of procedures. These procedures establish “how” to conduct the software validation effort. The procedures should identify the specific actions or sequence of actions that must be taken to complete individual validation activities, tasks, and work items.
- ***Software Validation after a Change***: Due to the complexity of software, a seemingly small local change may have a significant global system impact. When any change (even a small change) is made to the software, the validation status of the software needs to be re-established. Whenever software is changed, a validation analysis should be conducted not just for validation of the individual change, but also to determine the extent and impact of that change on the entire software system.
- ***Validation Coverage***: Validation coverage should be based on the software’s complexity and safety risk – not on firm size or resource constraints. The selection of validation activities, tasks, and work items should be commensurate with the complexity of the software design and the risk associated with the use of the software for the specified intended use.
- **Independence of Review**: Validation activities should be conducted using the basic quality assurance precept of “independence of review.” Self-validation is extremely difficult. When possible, an independent evaluation is always better, especially for higher risk applications.
- ***Flexibility and Responsibility***: Specific implementation of these software validation principles may be quite different from one application to another. Software is designed, developed, validated, and regulated in a wide spectrum of environments, and for a wide variety of devices with varying levels of risk.

4.3 EURECA validation planning

For EURECA, validation does not refer only to software components but also to processes (e.g. clinical scenarios). The main implications in clinical scenarios are that validation should cover all aspects of the process including the EURECA environment, any hardware that the environment uses, interfaces to other systems, the users, training and documentation as well as the management of the system and the validation itself after the system is put into use.

Section 2.1 summarizes the software components to be implemented (section 2.1.1) and the clinical scenarios (section 2.1.2). In order to assess the accuracy of the EURECA outcome, selected clinical scenarios will be described in detail, including the expected outcomes, and will be used as test cases.

The approach is to design an optional test cases list that is of reasonable size and can reveal as many errors existing in the system as possible. Actually, if test cases are selected randomly, many of these randomly selected test cases do not contribute to the significance of the EURECA platform, and thus, the number of random test cases is, not an indication of the effectiveness of the testing.

The test cases can have impact on different components, subsystems or the entire prototype depending on the specific requirement(s) they address.

In order to ensure that the user requirements are met, for each requirement there should be a detailed set of conditions which verify with certainty when a requirement has or has not been fulfilled.

A formal test case should include at least the following information:

- Preconditions. A set of input parameters and/or the state of the tested component(s) before a test is conducted.
- Post conditions. The expected result or effect of the test, in order for the tested component to pass or fail the test.

At this phase of the project a full and exhaustive list of test case is not possible to be defined.

Moreover, in EURECA we intend to perform an extensive evaluation of the clinical services offered by the infrastructure that will prove the EURECA project impact. The idea is that measurable parameters will be established in cooperation with the responsible clinical partners (e.g. recruitment rate, the number of SAE/SUSAR avoided etc.) for each clinical service offered within EURECA. Those measurable parameters will be monitored for a time frame [x1, x2] where the EURECA infrastructure is not used. Then, EURECA services will be used and the same parameter will be monitored. In this way we will be able to demonstrate the real impact of the EURECA infrastructure.

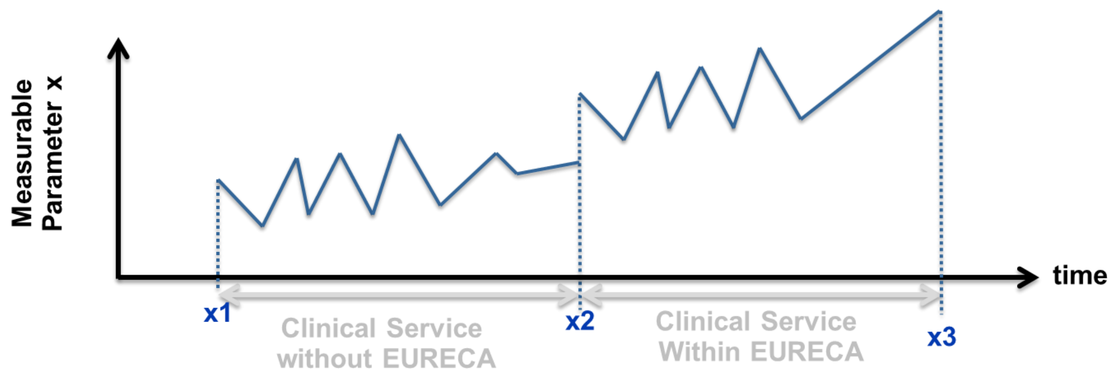

Figure 6: Measuring Parameter x before and after using EURECA infrastructure

4.3.1 Feed Back Results into the Loop

In line with the iterative approach, the validation results will contribute to the success of the project because all the user feedback will be shared with the software developers. To obtain feedback and validate the EURECA platform, we propose to use the spiral methodology. This will enable us to make updates and improvements to the system in more incremental steps. The spiral model (BW, 1988) uses the main processes of the more traditional waterfall method, requirements gathering, analysis, design and implementation, but all introduces the notion of an incremental process (see Figure 7).

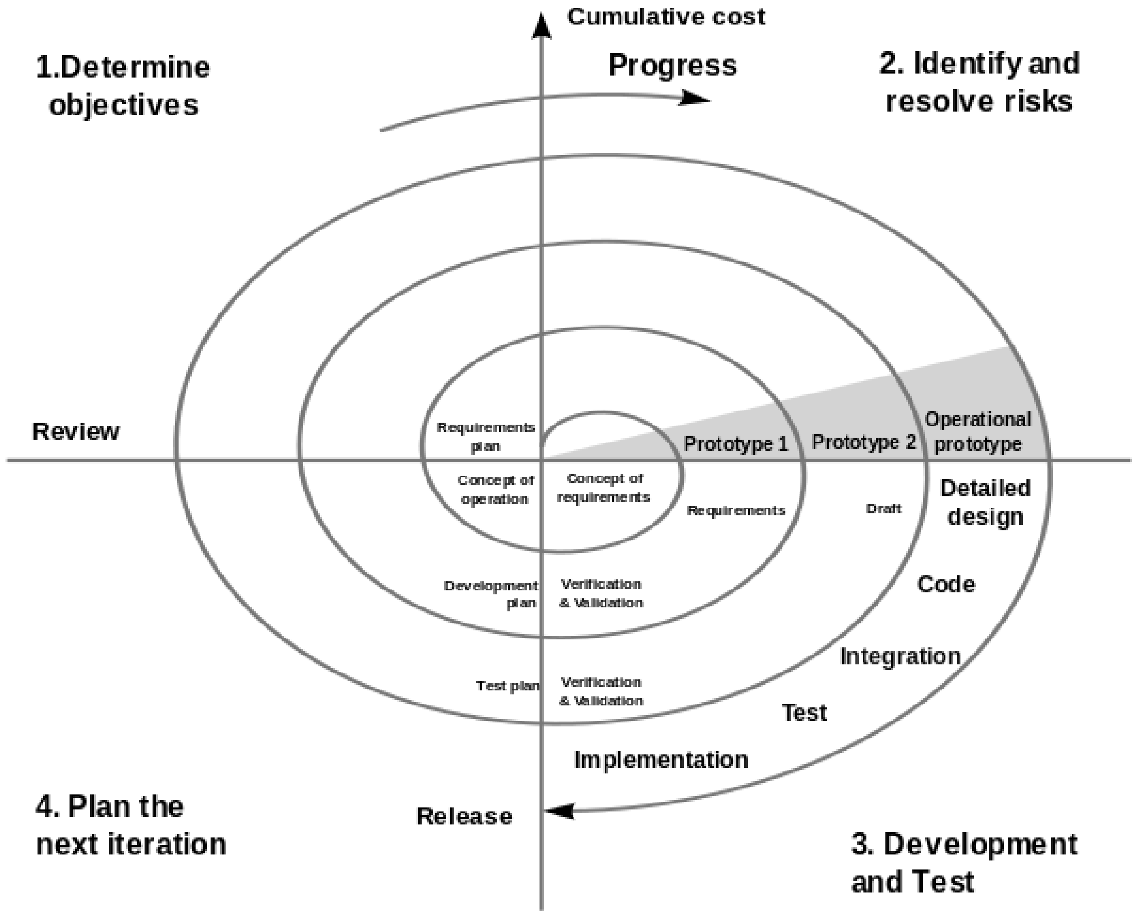

Figure 7: Spiral Development Model.

The spiral lifecycle model allows for elements of the product to be added in when they become available or known. This assures that there is no conflict with previous requirements and design. This method is consistent with approaches that have multiple software builds and releases and allows for making an orderly transition to a maintenance activity. Another positive aspect is that the spiral model forces early user involvement in the system development effort. For EURECA with heavy user interfacing such involvement is helpful.

Starting at the centre, each turn around the spiral goes through several task regions:

1. Determine the objectives, alternatives, and constraints on the new iteration.
2. Evaluate alternatives and identify and resolve risk issues.
3. Develop and verify the product for this iteration.

4. Plan the next iteration.

Designs and prototypes would be generated for the end users to use, validate and feedback on.

An example of such a process is as follows:

- The new system requirements are defined in as much detail as possible.
- A preliminary design is created for the new system.
- A first prototype of the new system is constructed from the preliminary design.
    - This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
- A second prototype is evolved using four steps:
    - Evaluate the first prototype and identify its strengths, weaknesses, and risks.
    - Define the requirements of the second prototype.
    - Plan and design the second prototype.
    - Construct and test the second prototype.
- Risk factors might involve development overruns, operating-cost miscalculation, or any other factor that could result in a less-than-satisfactory final product.
- The existing prototype is evaluated in the same manner as was the previous prototype, and, if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
- The preceding steps are iterated until the desired outcome is satisfied
- The final system is constructed, based on the refined prototype.

Such a process will take place many times as the EURECA environment grows to include different components and feedback from the end users.

While we are not able to state the exact criteria that we will use to measure the validation of the components and the test cases at the specific time point, we identified a general list with validation measures that we may use for the EURECA validation procedure. This list includes:

Quality measures:

Software quality is a multidimensional concept. The multiple professional views of product quality may be very different from popular or non-specialist views. Moreover, end users have levels of abstraction beyond even the viewpoints of the developer or user. However, very few end users will agree that a program that perfectly implements a flawed specification is a quality product. Typical criteria for quality measure are:

- ***Performance:*** Stakeholders have been measuring costs, quality, quantity, cycle time, efficiency, the cost in terms of time and other factors for carrying out the task. What is new to some extent is having those who the work determine some of what should be measured in order that they might better control, understand, and improve what they do. Effectiveness (the ability to actually carry out tasks successfully) is also measured under performance.
- ***Added Value:*** Adding value involves knowing what would be most useful for the software, how to communicate regularly with the users, following up, showing interest in future use

- ***Accuracy***: Accuracy is the degree to which data correctly reflects the real world object or event being described. Thus a software accuracy is how good the reflection of the real world comes to the user
- ***Acceptance of users:*** User acceptance can be defined as the demonstrable willingness within a user group to employ information technology for the tasks it is designed to support. Thus, acceptance measurement is less concerned with unintended uses or non-discretionary use of technologies and more interested in understanding the factors influencing the adoption of technologies as planned by users who have some degree of choice
- ***Subjective assessment (affect) of the quality of an application***: In general assessment can be objective or subjective.
  - Objective assessment is a form of questioning which has a single or multiple specific correct answers.
  - Subjective assessment is a form of questioning which may have more than one current answer (or more than one way of expressing the correct answer).
- ***Learning effort required using a system:*** Learning is acquiring new, or modifying existing, knowledge, behaviours, skills, values, or preferences and may involve synthesizing different types of information. The determined attempt to learn using a system could be measured with time.
- ***Cognitive workload:*** The notion of cognitive workload is ill-defined, it is often implicitly portrayed as something that cannot be reduced to a combination of more fundamental processes such as working memory load, attention, and so on.
- ***Security and Privacy***: Security is the degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization

Functionality measures:
In information technology, functionality is what the sum or any aspect of what a product, such as a software application or a service, can do for a user. A product's functionality is used by marketers to identify product features and enables a user to have a set of capabilities. Functionality may or may not be easy to use. Functionality testing is employed to verify whether the product meets the intended specifications and functional requirements. Typical criteria for functionality measure are:

- ***Completeness of necessary functionalities***: whether the system offers all necessary functionalities as well as how well the functionalities are implemented.
- ***Workflow support:*** The question if the functionalities support the usual workflow can be answered if we measure how well the everyday operation of the working environment is facilitated.
- ***Implementation of additional functionalities***: Functionality testing helps deliver software with a minimum amount of issues to an increasingly sophisticated pool of end users. Additional functionalities could be implemented after testing.

User satisfaction measures:

User satisfaction is difficult to measure for several reasons. One has to count on users not only to give feedback, but also to be honest in their assessment, as well as provide surveys in several ways (through mail, email, or over the phone) and in order to get the best information to allow customers to answer questions on a weighted scale. The questions of surveys should be designed as to be able to help draw conclusions on time to complete task as well as completion rate percentage, repetitions of failed commands, misleads of user interface, disruption of work task, system control loss etc. The factors, which might affect user satisfaction with, are contained in this parameter list. It is important that for each parameter in the list satisfaction should be quantifiable. Some quantification measures are easily defined. The quantification may be defined as an integer value. Other parameters may have more subjective quantifications. This list of measures identifies several alternative metrics.

The metrics are intended to measure user performance only, while deliberately ignoring user satisfaction and design elements that are or are not visible to the user:

- Time to complete task
- Percentage of task completed
- Percentage of task completed per unit time (speed metric)
- Ratio of success to failures
- Time spent on errors
- Percentage number of errors
- Number of commands used
- Frequency of help or documentation use
- Time spent using documentation
- Percentage of favourable / unfavourable comments
- Number of repetitions of failed commands
- Number of times the interface misleads the user
- Number of good and bad features recalled by the user
- Number of available commands not invoked
- Number of regressive behaviours
- Numbers of times users need to work around a problem
- Number of times the user is disrupted from a work task
- Number of times the user loses control of the system
- Number of times the user expresses frustration or satisfaction

Design reviews during the early development phase should be carried out by the project’s experts, which are not involved in the development effort (clinical partners). The objective is to use checklists and test the system according to the defined test cases, assuming the role of a user. Finally the results will be reported directly to the developers, and possibly involve the developers in the design review.

The test cases for the validation will be described in detail in the deliverable (D8.2) along with the validation procedures and the exact measurement criteria that will be used.

5 Conclusions

This document established the requirements of the evaluation, identified the products to be evaluated and identified the measures and models for the evaluation. As such, the first two blocks from Figure 1 were defined, i.e. the “Establish evaluation requirements” and “Specify the evaluation” blocks.

The remaining components of the complete evaluation model (based on the ISO/IEC 25040 and adapted for use in EURECA) will be executed and reported in subsequent activities of the project.
Specifically, the “Design the evaluation” block will be reported in “ (D8.2)- Specifications of the evaluation and validation scenarios for the different EURECA components” and “ (D8.4) – Specifications of the evaluation and validation scenarios and demonstrators for the clinical pilots” whereas the “Execute the evaluation” and “Conclude the evaluation” blocks will be reported in “ (D8.3) - Report on evaluation and validation of EURECA components”, “ (D8.5) - Report on the evaluation and validation of the EURECA environment and services” and “ (D8.6) - Report on the user workshops at clinical sites”

6 Bibliography

2500n, I. (n.d.). Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE.

25010, I. (n.d.). Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models.

25020, I. (n.d.). Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Measurement reference model and guide.

25030, I. (n.d.). Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Quality requirements.

25040, I. (n.d.). Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- Evaluation process.

BW, B. (1988). A Spiral Model of Software Development and Enhancement. *Computer*, 61-72.

D1.1, E. (n.d.). User needs and specifications for the EURECA environment and software services .

D1.2, E. (n.d.). Definition of relevant user scenarios based on input from users.

D2.2, E. (n.d.). D2.2 – Initial EURECA architecture.

D8.2, E. (n.d.). Specifications of the evaluation and validation scenarios for the different EURECA components.

D8.3, E. (n.d.). Report on evaluation and validation of EURECA components.

D8.4, E. (n.d.). Specifications of the evaluation and validation scenarios and demonstrators for the clinical pilots.

D8.5, E. (n.d.). Report on the evaluation and validation of the EURECA environment and services.

D8.6, E. (n.d.). Report on the user workshops at clinical sites.

FDA. (1997, 6 9). General Principles of Software Validation; Final Guidance for Industry and FDA Staff. *General Principles of Software Validation, Version 1.1*.

IEC. (n.d.). *http://www.iec.ch/*.

ISO. (n.d.). *http://www.iso.org*.

ISO/IEC12207:2008. (n.d.). Systems and software engineering -- Software life cycle processes.

ISO8402:1994. (n.d.). Quality management and quality assurance.

SQuaRE, I. 2. (n.d.). Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) – Evaluation reference model and guide.

Appendix A

Eureca software evaluation form

Name of evaluator(s):						
Evaluator's expertise:						
Name and Version of the EURECA software component:						
Evaluation date :						
		Rating				
		strongly agree	agree	neutral	disagree	strongly disagree
Functionality	The set of functions covers all the specified tasks and user objectives.					
	The system provides the correct results with the needed degree of precision.					
	The functions facilitate the accomplishment of specified tasks and objectives.					
Efficiency	The system responds quickly.					
	The system utilizes resources efficiently.					
Compatibility	The system shares resources without loss of its functionality.					

	The system shares information/data with other EURECA components?					
Usability	The users can recognize easily whether the system is appropriate for their needs.					
	The users learn to use the system easily.					
	The users use the system without much effort.					
	The system protects users against making errors.					
	The user interface enables pleasing and satisfying interaction for the users.					
Reliability	Most of the faults in the software been eliminated over time.					
	The software is capable of handling errors.					
	The software resumes working & restores lost data after failure.					
Security	The system provides identification access wherever is needed.					
	Data are accessible only to authorized users					
	The system traces actions uniquely.					
	The system prevents unauthorized access.					
ntai nab	Faults can be easily diagnosed.					

	The system is composed of discrete independent components.					
	An asset can be used in more than one system, or in building other assets.					
	The software can be tested easily.					
Portability	The software can be moved to other environments easily.					
	The software can be installed easily.					
	The software can easily replace other software.					
Quality of use	The software is accurate and complete for the intended use.					
	The software improves the time or reduces resources for the intended goal.					
	The software satisfies the perceived achievements of pragmatic goals.					
	The software cannot harm people in the intended contexts of use.					