D6.1

Basic validation methodology and Quality control indicators

<table>
<thead>
<tr>
<th>Editor:</th>
<th>Franck Le Gall, Easy Global Market</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deliverable nature:</td>
<td>Report (R)</td>
</tr>
<tr>
<td>Dissemination level:</td>
<td>Public (PU)</td>
</tr>
<tr>
<td>Contractual delivery</td>
<td>M6 (M9)</td>
</tr>
<tr>
<td>Actual delivery date:</td>
<td>M10</td>
</tr>
<tr>
<td>Suggested readers:</td>
<td>All FI-STAR partners. Participants to FI-PPP projects willing to develop KPIs based quality assessment and planning</td>
</tr>
<tr>
<td>Version:</td>
<td>1.0</td>
</tr>
<tr>
<td>Total number of pages:</td>
<td>94</td>
</tr>
<tr>
<td>Keywords:</td>
<td>Quality assessment framework, label, Key Performance indicators, model based testing, health impact</td>
</tr>
</tbody>
</table>
Abstract

Deliverable D6.1 of project FI-STAR describes the “Basic validation methodology and Quality control indicators” of the project. It positions itself in the global framework of the FI-PPP programme and defines quality indicators along the dimensions of sectorial (health) impact, software quality, quality of experience and quality of service, allowing in addition to evaluate sustainability of activities pursued by the project as well as outcomes from FI-PPP used by the project (e.g. Generic enablers).

Advanced tools and methods are proposed for the collection of indicators. These range from qualitative approaches for end-users appreciation to Model Based Testing for conformance testing and system performance evaluation.

Standards based processes for teams’ interactions are described.

[End of abstract]
Disclaimer

This document contains material, which is the copyright of certain FI-STAR consortium parties, and may not be reproduced or copied without permission.

All FI-STAR consortium parties have agreed to full publication of this document.

The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the FI-STAR consortium as a whole, nor a certain part of the FI-STAR consortium, warrant that the information contained in this document is capable of use, nor that use of the information is free from risk, accepting no liability for loss or damage suffered by any person using this information.

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 609094

Impressum

[Full project title] Future Internet Social and Technological Alignment Research
[Short project title] FI-STAR
[Number and title of work-package] WP6 “Quality and Validation”
[Number and title of task] T6.1 “Validation methodologies & tools”
[Document title] D6.1 “Basic validation methodology and Quality control indicators”
[Editor: Name, company] Franck Le Gall, Easy Global Market
[Work-package leader: Name, company] Franck Le Gall, Easy Global Market

Copyright notice

© 2013 Participants in project FI-STAR

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-nd/3.0
Executive summary

The workpackage 6 (WP6) “Quality and Validation” of FI-STAR establishes quality monitoring processes within the FI-STAR project. The present document describes the first step being to provide a quality assessment framework including overall methodology, tools and indicators to the project as how to ensure conformity to project objectives and results.

The scope of the quality assessment framework is firstly clarified by identifying:

- **The interested parties (Stakeholders):** persons interested to make use or take benefit from the quality framework and having diverse quality expectations. These include the intended beneficiaries of FI-STAR outcomes, the project partners (with a specific focus on software developers), FI-PPP projects and steering committee as well as policy makers such as the European commission.

- **The subjects to be evaluated:** These include both generic and specific enablers, the platforms instantiating these enablers and the applications running on top of these platforms.

- **The Dimensions:** Four dimensions have been retained covering the field of eHealth (specificities of the eHealth sector, including expected added value), Software quality (verification and validation – including compliance testing, interoperability, and reusability), Quality of Experience (perceived from either the end users/beneficiaries and the developers) and Quality of Service (evaluating the end to end service delivery performance).

The WP6 works in close relation with other workpackages in the project so fostering the use of a shared set of Key Performance Indicators (KPIs) among involved parties. The present document focuses on indicators being linked or of interest to third parties to the project. Any indicator solely related to the internal project performance is managed directly within the management activities of the project and is thus not considered in the present document.

The third chapter of the report provides details the approaches taken to select indicators within each of the retained dimensions. Selection has been made with the objective to overall retain a limited number of indicators (30 to 40) thus keeping affordability within the project resources and clarity when communicating to third parties. Whenever possible, existing and robust evaluation frameworks have been used as a basis for the indicators selection. Worthwhile to be noted are the use of:

1. MAST (Model for A5ssessment of Telemedicine) mapped to evaluate the impact of FI-STAR services on use case objectives. This method allowed to select 10 areas of main interest for health impact evaluation in FI-STAR: accessibility, adhereability, affordability, authenticity, availability, efficiency, effectiveness, empowerment, safety and trustability.

2. Reusability Readiness Levels (RRL) is an extension of the Technology Readiness Levels (TRL) scale produce by NASA and used by the European Commission in the context of H2020. RRL produces a scale to evaluation reusability of software components by looking at: documentation, extensibility, intellectual property,
modularity, packaging, portability, standards compliance, support and verification and testing

Then the fourth section provides details the advanced tools to be used to bring the quality assessment framework to life, especially in respect with software quality evaluation. Within this section, accent is made on:

1. The implementation of the IEC 62304:2006 standard “Medical device software -- Software life cycle processes” as a guideline to monitor software development processes within the project

2. The use of model based testing approaches in which a test model, representing a (semi) formal representation of the system from a testing perspective, is used. This participates to conformance testing of enablers against their specification as well as applications requirement coverage evaluation.

Finally, the detailed list of KPIs is provided, including definition of the indicator, scales boundaries, caveats, contributors and updates frequency. By default target are proposed for each of these KPIs ad propagated over the subjects intended to be evaluated.

The table belows provide an overview of the subjects to be evaluated with the proposed quality assessment framework

<table>
<thead>
<tr>
<th>FI-PPP programme</th>
<th>Software quality</th>
<th>QoS</th>
<th>QoE</th>
<th>e-Health</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic enablers</td>
<td>✓ Reusability</td>
<td>✓ QoS</td>
<td>✓ QoE (dev.)</td>
<td>✓ Interoperability</td>
</tr>
<tr>
<td>Platforms (testbeds)</td>
<td>✓ QoS</td>
<td>✓ QoS</td>
<td>✓ QoE (dev.)</td>
<td>✓ Interoperability</td>
</tr>
<tr>
<td>Specific enablers</td>
<td>✓ Verification</td>
<td>✓ Interoperability</td>
<td>✓ Reusability</td>
<td>✓ Interoperability</td>
</tr>
<tr>
<td>FI-STAR project</td>
<td>✓ QoS</td>
<td>✓ QoE (dev.)</td>
<td>✓ Interoperability</td>
<td></td>
</tr>
<tr>
<td>Use case Platforms</td>
<td>✓ QoS</td>
<td>✓ QoE (user)</td>
<td>✓ Ethical &amp; Social</td>
<td></td>
</tr>
<tr>
<td>Applications</td>
<td>✓ Verification</td>
<td>✓ QoS</td>
<td>✓ QoE (user)</td>
<td>✓ Health impact</td>
</tr>
<tr>
<td>3rd parties</td>
<td>✓ Verification</td>
<td>✓ QoS</td>
<td>✓ QoE (user)</td>
<td>✓ Interoperability</td>
</tr>
<tr>
<td>Sensors/Actuators</td>
<td>✓ QoS</td>
<td>✓ QoE (user)</td>
<td>✓ Interoperability</td>
<td></td>
</tr>
<tr>
<td>Infrastructure</td>
<td>✓ QoS</td>
<td>✓ QoE (user)</td>
<td>✓ Interoperability</td>
<td></td>
</tr>
</tbody>
</table>
List of authors

<table>
<thead>
<tr>
<th>Company</th>
<th>Author</th>
<th>Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy Global Market (EGM)</td>
<td>Philippe Cousin</td>
<td>Document creation Methods for MBT (§4.2)</td>
</tr>
<tr>
<td>Easy Global Market (EGM)</td>
<td>Franck Le Gall</td>
<td>Overall edition. Overall scope (§2, §3.1) Reusability methods (§3.4.2, §5.3.3, Annex B)</td>
</tr>
<tr>
<td>Blekinge Tekniska Hogskola (BTH)</td>
<td>Samuel Fricker</td>
<td>Support to overall edition Verification (§3.4.1, §5.3.1) Internal interoperability (§5.3.2) Overall process (§4.1)</td>
</tr>
<tr>
<td>Blekinge Tekniska Hogskola (BTH)</td>
<td>Markus Fiedler</td>
<td>QoE (§3.2, §5.1, Annex A) QoS (§3.3, §5.2)</td>
</tr>
<tr>
<td>Blekinge Tekniska Hogskola (BTH)</td>
<td>Peter Anderberg</td>
<td>eHealth impact (§3.5.1, §5.4.2)</td>
</tr>
<tr>
<td>Blekinge Tekniska Hogskola (BTH)</td>
<td>Shahryar Eivazzadeh</td>
<td>eHealth impact (§3.5.1, §5.4.2)</td>
</tr>
<tr>
<td>Stichting Katholieke Universiteit Brabant Universiteit Van Tilburg (TiU)</td>
<td>Nadya Purtova</td>
<td>Privacy impact assessment (§3.5.2)</td>
</tr>
<tr>
<td>Stichting Katholieke Universiteit Brabant Universiteit Van Tilburg (TiU)</td>
<td>Federica Lucivero</td>
<td>Ethics and Social (§3.5.1, §5.4.1)</td>
</tr>
<tr>
<td>Elvior (ELV)</td>
<td>Dean Felmly</td>
<td>Model based testing (§4.2)</td>
</tr>
</tbody>
</table>

In addition, participants from WP2 to WP2 have been consulted and participated in the identification and selection of KPIs.
Table of Contents

Executive summary ...................................................................................................................................... 4
List of authors .......................................................................................................................................... 6
Table of Contents ..................................................................................................................................... 7
List of figures ............................................................................................................................................ 10
List of tables ........................................................................................................................................... 10
Abbreviations .......................................................................................................................................... 11
1 Introduction ........................................................................................................................................ 12
2 Scope of the quality assessment framework ....................................................................................... 15
  2.1 Overview ......................................................................................................................................... 15
  2.2 Interested stakeholders .................................................................................................................... 16
  2.3 Targets to be considered ................................................................................................................. 17
  2.4 Dimensions of interest ...................................................................................................................... 18
  2.5 FI-STAR activities and integration of quality activities .................................................................... 20
3 Validation methodologies ....................................................................................................................... 23
  3.1 Context: Overall FI-STAR software development process ............................................................ 23
  3.2 Quality of experience (QoE) .......................................................................................................... 24
    3.2.1 Generic view ............................................................................................................................. 25
    3.2.2 User view .................................................................................................................................. 25
    3.2.3 Developer view ......................................................................................................................... 26
    3.2.4 QoE Evaluation Protocol ......................................................................................................... 26
  3.3 Quality of Service (QoS) .................................................................................................................. 27
    3.3.1 Instrumentation of response time measurements .................................................................. 28
    3.3.2 Instrumentation of error measurements ................................................................................. 30
    3.3.3 Instrumentation of availability measurements ....................................................................... 30
  3.4 Software quality ............................................................................................................................... 30
    3.4.1 Verification and Validation (V&V) ......................................................................................... 30
    3.4.2 Reusability ............................................................................................................................... 32
  3.5 eHealth ............................................................................................................................................ 37
    3.5.1 E-Health Impact ....................................................................................................................... 37
    3.5.2 Privacy Impact Assessment as a quality indicator ................................................................. 40
    3.5.3 Health interoperability ............................................................................................................. 46
4 Processes and methods for software quality ....................................................................................... 47
4.1 Overall process for quality monitoring ......................................................... 47
  4.1.1 Context ...................................................................................................... 47
  4.1.2 Quality Monitoring Principles .................................................................. 48
  4.1.3 Process and Responsibilities ..................................................................... 50
4.2 Methods for Model Based formal conformance testing ................................. 52
  4.2.1 Black-box test automation ......................................................................... 52
  4.2.2 Model based testing overview .................................................................... 53
  4.2.3 Taxonomy of model-based testing .............................................................. 56
  4.2.4 Overview of MBT tools ............................................................................. 59
  4.2.5 MBT approach proposed for FI-STAR ...................................................... 60
  4.2.6 Data collection and analysis ..................................................................... 64
5 Quality indicators ............................................................................................... 65
  5.1 Quality of experience .................................................................................... 65
  5.2 Quality of service .......................................................................................... 67
  5.3 Software quality ............................................................................................. 68
    5.3.1 Verification ............................................................................................... 68
    5.3.2 Internal interoperability .......................................................................... 69
    5.3.3 Reusability ............................................................................................... 69
  5.4 E-health .......................................................................................................... 70
    5.4.1 Ethical and Social .................................................................................... 70
    5.4.2 Health impact .......................................................................................... 71
    5.4.3 eHealth interoperability ......................................................................... 72
  5.5 Quality targets ................................................................................................. 73
6 References ........................................................................................................... 78
Annex A QoE Evaluation Questionnaire ................................................................. 83
Annex B Detailed explanation of the Reuse readiness scale ................................... 85
  B.1 Documentation ............................................................................................... 85
  B.2 Extensibility ................................................................................................... 86
  B.3 Intellectual Property Issues .......................................................................... 87
  B.4 Modularity ...................................................................................................... 89
  B.5 Packaging ....................................................................................................... 89
  B.6 Portability ........................................................................................................ 90
  B.7 Standards Compliance .................................................................................. 91
  B.8 Support ........................................................................................................... 92
B.9 Verification and Testing
List of figures

Figure 1: FI-PPP program .............................................................. 13
Figure 2: overview of FI-STAR validation tools ........................................ 16
Figure 3: stakeholders identified in the quality assessment framework ............... 17
Figure 4: potential subjects to be evaluated within the quality assessment framework. 18
Figure 5: Main thematic areas of FI-STAR quality assessment framework .......... 20
Figure 6: FI-STAR quality assessment labelling model ................................ 21
Figure 7: FI-STAR software verification and validation activities ....................... 31
Figure 8: Decision tree on whether and at what level of details to conduct a PIA (source ................................................. 42
Figure 9: Usage relationships between healthcare ecosystem, FI-STAR applications, and FI-STAR/FI-PPP platforms. Each enabler, application, and use scenario has an owning development, respectively experimentation team. ........................................ 47
Figure 10: The Model-Based Testing Process ........................................ 53
Figure 11: MBT workflow ................................................................ 55
Figure 12: Overview of the taxonomy for MBT in [48] ..................................... 56
Figure 13: From specification documents to executable TTCN-3 test cases .......... 61
Figure 14: from TTCN-3 test cases to executable tests .................................. 62
Figure 15: Building a UML test model .................................................. 63

List of tables

Table 1: focus of quality evaluation activities .............................................. 22
Table 2: QoE evaluation workshop process ................................................ 27
Table 3: Reuse Readiness Level (RRL) Topic Area Levels Summary (based on [23]) .... 35
Table 4: IEC 62304 Process Requirements for Development of Class A Applications and Enablers ......................................................... 48
Table 5: IEC 62304 Process Requirements for Maintenance of Class A Applications and Enablers .......................................................... 49
Table 6: Triggering of V&V and quality monitoring activities (facilitated by WP6 if not indicated otherwise) ...................................................... 51
Table 7: Quality Monitoring Responsibilities ............................................. 52
Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FI-PPP</td>
<td>Future Internet Public-Private Partnership</td>
</tr>
<tr>
<td>MAST</td>
<td>Model for ASsessment of Telemedicine</td>
</tr>
<tr>
<td>MBT</td>
<td>Model Based Testing</td>
</tr>
<tr>
<td>M2M</td>
<td>Machine to Machine</td>
</tr>
<tr>
<td>PPI</td>
<td>Project performance indicators</td>
</tr>
<tr>
<td>QoE</td>
<td>Quality of Experience</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>QPI</td>
<td>Quality performance indicators</td>
</tr>
<tr>
<td>RRL</td>
<td>Reuse readiness level</td>
</tr>
<tr>
<td>V&amp;V</td>
<td>Verification and Validation</td>
</tr>
</tbody>
</table>
1 Introduction

With over a billion users world-wide, the Internet is one of history’s great success stories. Its global, integrated communications infrastructures and service platforms underpin the fabric of the European economy and society. Yet today’s Internet was designed in the 1970s, for purposes that bear little resemblance to current and future usage scenarios. Mismatches between the original design goals and how the Internet is being used are beginning to hamper its potential. Many challenges in the areas of technology, business, society and governance will have to be overcome if the future development of the Internet is to sustain the networked society of tomorrow.

To answer these challenges, the European Commission has launched the Future Internet Public-Private Partnership Programme (FI-PPP). The main goal is to advance a shared vision for harmonised European-scale technology platforms and their implementation, as well as the integration and harmonisation of the relevant policy, legal, political and regulatory frameworks. As set forth in the Digital Agenda for Europe, these are considered to be prerequisites for realizing a European online Digital Single Market (DSM) and, more broadly, an inclusive knowledge society.

Programme aims are to:

- Increase the effectiveness of business processes and infrastructures supporting applications in areas such as transport, health, and energy.
- Derive innovative business models that strengthen the competitive position of European industry in sectors such as telecommunication, mobile devices, software and services, and content provision and media.

The FI-PPP follows an industry-driven, holistic approach encompassing R&D on network and communication infrastructures, devices, software, service and media technologies;

In parallel, it promotes their experimentation and validation in real application contexts, bringing together demand and supply and involving users early in the research lifecycle.

The new platform will thus be used by a range actors, in particular SMEs and Public Administrations, to validate the technologies in the context of smart applications and their ability to support «user driven» innovation schemes.
The participation of actors including SMEs and Public Administrations, to validate the technologies will lead to involve much more actors to use the platform components and therefore will raise a lot of new issues related to support these new stakeholders in using the FI-PPP components. **Confidence in the whole programme will be at stake and can be easily endanger if too much weaknesses would be detected.** Information on difficulties to use the platform or weaknesses in technical elements uses to be quickly spread up by communities. To prepare such important taking up by larger communities, **it is important to develop quality programme to reinforce the robustness of the technical elements while giving reasonable level of confidence to users.** This kind of activities are the basis for the establishment of certification or label scheme and such concept was raised at FI-PPP level in particular when FI-STAR made clear on its firm intention for its own project to ensure robustness of its development and to give confidence to end users by certification.

In such new expectation from the FI-PPP programme, FI-STAR is invited to provide input to “certification” where good practices might be extended to the whole programme. As a part of FI-STAR, but not only, will be dedicated to **check conformity of GEs,** it is likely feasible that FI-STAR tests and tools could be used for the FI-PPP programme.

EC made also clear that, while FI-STAR is operating in the eHealth domain, one of its important objectives is, through the eHealth demonstrators and pilots, to support and validate the whole programme. Such statement made clear to FI-STAR partners that looking at certification requirements means paying a large attention to certification supporting the FI-PPP programme. Therefore high attention will be devoted to the possible use of certification approaches as part of the quality process.

The workpackage 6 (WP6) “Quality and Validation” of FI-STAR aims at establishing quality monitoring processes within the FI-STAR project. It provides overall methodology, tools and indicators to the project as how to ensure conformity to project objectives and results.
Developing a quality assessment framework requires to first clearly define its scope in terms of:

- **Stakeholders**: who are the groups of persons involved in the use of the quality assessment framework
- **Subjects**: what are the subjects\(^1\) of which quality is to be evaluated
- **Dimensions**: what are the topics to be considered in the quality evaluation and to which metrics\(^2\) can be associated. The framework associates indicators to the metrics thus allowing to get a measure of the quality level against the different considered dimensions.

The next section further details these categories.

It is worthwhile noting that the purpose of the quality assessment framework is not to evaluate everything in all dimensions but rather to focus on few points of interest expressed by the stakeholders and for which metrics can be collected. For that purpose, attention will be paid in selecting S.M.A.R.T. (Specific, Measurable, Assignable, Realistic and Time bounded) indicators.

The WP6 works in close relation with other workpackages in the project so fostering the use of a shared set of Key Performance Indicators (KPIs) among involved parties. The WP6 (and thus the present document) focuses on indicators being linked or of interest of third parties to the project. Any indicator solely related to the internal project performance is managed directly within the management activities of the project and is thus not considered in the present document. The distinction is then made between:

- **Quality performance indicators (QPI)**: related to interaction with project third party (providers of generic enablers, users and beneficiaries of the FI-STAR applications, etc.)
- **Project performance indicators (PPI)**: solely related to internal project performance

---

\(^1\) Can be either material (such as hardware) or immaterial (such as software, process, etc.)

\(^2\) In this document, metrics can refer to either quantitative or qualitative information. The wordings pseudo-quantitative or semi-quantitative can be used to indicate qualitative evaluations made against closed scales such as Likert scales or Booleans.
2 Scope of the quality assessment framework

2.1 Overview

A study was first undertaken to define the requirements on standardisation and certification the project would have to comply with. This includes in particular the outlining of a validation and certification strategy for FI-STAR, which provides easy and simple proof of compliance to requirements to the market. This strategy needs to be pragmatic while complying with the existing regulatory framework, helping the issuing of high quality and sustainable outcomes by the project FI-STAR and the FI-PPP programme.

(Three different approaches are then proposed depending on the level of the considered requirements:

a) Regulatory requirements: there is no other choice than ensuring the regulatory compliance of systems, devices and applications put on the market by the project FI-STAR. Nevertheless, this is usual business of the health IT providers being present in the project and thus does not require project specific attention.

b) External requirements: these requirements comes from third parties, external to the project and does not contain mandatory requirements of standardization or certification of IT products or IT-based services. Motivation for certification is thus mostly guided by interoperability interests (also here as also promoted as interchangeability and plugability): FI-STAR developers want to benefit from components (GEs) from the FI-PPP that are in-line with their specification while FI-STAR wishes to develop platforms that are re-useable by third parties (SEs) and which can connect and exchange data with medical systems. In that case, 2 options exist:

- Validation of GEs/SEs: this is not provided by any third party today and thus has to be conducted at the FI-STAR level and promoted at the FI-PPP level. This activity is being driven with WP6.

- Interoperability with health systems: many profiles based interoperability schemes and tools are already promoted by many EU projects and FI-STAR has to capitalize on this important existing offers. It should be conducted on a voluntary basis. For instance Projectathon organised in the framework of connectathon is one of the most efficient way to expose FI-STAR outcome to the health market reality. Such activity has to be done by development teams within FI-STAR with the support of dissemination and exploitation teams. Other offers are developed within the eHealth thematic network Antilope.

a) Internal requirements: while we need to communicate with the external world, we need to be sure what we deliver has high quality and is conform to the expectation (also from users’ side). Quality improvement has to be sought to increase the acceptability of FI-STAR outcomes. The project thus has to set its own quality requirements and deploy its internal processes to monitor them.

To cover all the validation requirements we will use several methodologies such as

- V&V White Box testing (e.g. using IEEE1012 [1]) for addressing the Internal requirements
- V&V Black-Box testing for conformity to GEs or SEs specifications
- Reusability Readiness Level (RRL)
- Quality of Experience (QoE) and Quality of Services (QoS)
- Model for ASsessment of Telemedicine (MAST) for evaluation of health impacts;

These methods are further described in this report.

![Figure 2: overview of FI-STAR validation tools](image)

### 2.2 Interested stakeholders

Thanks to the value chain analysis presented in section 2.1, five stakeholders categories are defined, based on their differentiated expectations and interactions with the FI-STAR project. These categories are (Figure 3):

- **Beneficiaries/end-users**: these are the one which ultimately use and benefit from FI-STAR applications. They are mainly the patients, expecting to get a higher medical value/cost ratio.

- **Medical care**: these are the one delivering the care. It includes doctors, nurses and more generally the health sector. As beneficiaries, they wish to increase the medical value/cost ratio of the services they provide. In addition, they need to ensure sustainability of these services.

- **FI-STAR developers**: these are the people in charge of the development of FI-STAR specific enablers (SE), platforms and applications. Different teams are involved in each step and each expect:
  - To increase the effectiveness and efficiency of their developments by the use of generic (provided by the FI-PPP core platform) or specific (provided by FI-STAR) enablers.
To deliver enablers, platforms or application performing as set in the requirements

- **FI-PPP platforms**: specific enablers developed by FI-STAR are expected to be reused by external platforms. The platforms providers, as users of the specific enablers, will have additional requirements to ensure they can deploy the specific enablers in a sustainable way.

- **European Commission**: the European Commission is the policy maker co-funding the FI-PPP program. Its high level objectives are to “increase the effectiveness of business processes and of the operation of infrastructures supporting applications in sectors such as transport, health, or energy; as well as to derive possible innovative business models in these sectors, strengthening the competitive position of European industry in domains like telecommunication, mobile devices, software and service industries, content providers and media” [2]. In that perspective it expects to get a measure of the efficiency and effectiveness of the program outcome, focused mostly on the enablers in the phase II to which FI-STAR belongs.

![Stakeholders](image)

**Figure 3**: stakeholders identified in the quality assessment framework.

### 2.3 Targets to be considered

Quality expectations also depend on the subject evaluated using the quality assessment framework. The dimensions of interest may be different, the methods and tools to collect them may vary as well as the expected quality level targets. It is thus necessary to identify the categories of subjects to be evaluated using the quality assessment framework so to:

- **Focus on subjects of interest**: categories of subjects not to be evaluated, or to be evaluated partially, need to be rapidly identified so to focus the effort where relevant
b) Adapt the approach: as said above, each subject of interest may be associated with different quality level expectations and tools and methods to measure them, may vary.

Categories of subjects have been identified at the levels of FI-PPP programme overall, at the FI-STAR project level and at the level of third parties.

- **At the FI-PPP level**, we find the Generic Enablers produced by the core platform and made available through the enablers catalogue [3]. Instances of these enablers can be run “As A Service” from platforms managed within the FI-PPP: Open Innovation Lab (OIL) [4] and Xifi (federation of platforms) [5].

- **At the FI-STAR project level**, which is the focus of the quality assessment, it starts from the Specific Enablers foreseen to be either built from Generic Enablers composition or completely developed depending on the correspondence between project requirements and functionalities offered by the GEs. Then these enablers are instantiated within platforms which are then linked through public or private cloud services. Project application are run from these infrastructures.

- **At third party level**: here are considered elements used in the deployment of FI-STAR solution but not under the project direct control. Examples of such elements include sensors and medical devices used as part of the applications as well as legacy systems of the involved organisations.

Figure 4: potential subjects to be evaluated within the quality assessment framework.

### 2.4 Dimensions of interest

Selection of the dimensions to be included in the quality assessment framework have been done over a period of 3 months which allowed to go first through a phase of expansion during which all suggestions arising from the project where included. This collection of domains of interest from the project description of work, the workpackages’ activities as well as expertise of WP6 contributors allowed to identify the points of interest that should likely be included in the quality assessment framework and more importantly: the points of interactions between involved parties.

The second phase of selection provided a focused and coherent set of domains, together with their corresponding KPIs through the application of several guiding principles:
The number of domains of interest and related KPIs had to remain low (3 to 5 domains; 30 to 50 KPIs) so to be manageable within the project resources and be easily communicable outside of the quality assessment team.

Redundancies and gaps among the indicators had to be avoided. Whenever possible, the selection of indicators had to rely on existing standards and methods and their evaluation be feasible through existing tools.

Ultimately, four dimensions of interest have been retained as part of the quality assessment framework (Figure 5). These dimensions where the characterized into sub-dimensions and then indicators. These dimensions are:

- **eHealth**: This dimensions includes aspects related to the specificities of the eHealth sector and evaluate the value delivered according to the value cases defined in D1.1 Technical Requirements (Health impact) as well as efforts to comply with EU practices related to interoperability in the field of eHealth (eHealth interoperability).

- **Software quality**: This part focuses on verification and validation of used (generic enablers) and produced (specific enablers and applications) software against its intended use but also analyse its reusability. Approaches proposed include in particular the integration and interoperability evaluation (enablers and 3rd-party systems/components/devices should interoperate according to predefined profiles), compliance (software should comply with the regulations and standards related to the health sector) and coverage (software that is put into use should be completely tested. Particular testing rigour is applied to critical parts of the system).

- **Quality of Experience (QoE)**: quality of experience reflects the perception from a user of service. ‘User’ has to be understood in the wide sense: users include the end users has well as the medical teams intended to be the beneficiaries of the FI-STAR applications and services. In addition, software developers of the project are themselves users of pieces of software developed by other teams either from the FI-STAR project or from external stakeholders. Analysis will then differentiate Quality of Developer Experience (Developer QoE, looking in particular at the extent to which enablers are accepted by developers and deliver value superior to alternative 3rd-party development frameworks or components) to the quality of user experience (beneficiary QoE).

- **Quality of service (QoS)**: aimed at evaluating the end to end service delivery quality and correlating it with the end users quality of experience.

Finally the sustainability of the FI-STAR, and more largely speaking Fi-PPP, has been retained as a dimension of high interest but which corresponding indicators are already included in the other dimensions, in particular the one being sector specific (health impact) and the one related to reusability of the software.
2.5 FI-STAR activities and integration of quality activities

WP6 delivers a systematic evaluation and testing process that integrates into the overall FI-STAR development process. Its principles are based on widely-established good industry practice [6] and regulatory requirements as defined in D1.2 [7].

The project FI-STAR is made of 9 workpackages (WP):

- WP1 specifies usage specific and generic requirements from the perspective of the different use case scenarios and experimentation sites, including the overall architecture, and drives developments in a coherent way.
- WP2 and WP3 are responsible for the development, validation and test of the infrastructure services components for the FI-STAR service providers and consumers communities.
- WP4 is responsible for the development, validation and test of a new generation of ultra-light FI-applications aiming at the usage specific requirements.
- WP5 is responsible for the deployment, integration, test of the FI-STAR products (delivered by WP2-4) on the experimentation site operational facilities of the FI-STAR service providers and consumers and makes these available through trial executions. FI-STAR follows an iterative approach, with two development and two experimentation cycles
- WP6 is responsible for the evaluation of the FI-STAR system from the perspective of usability, performance, cost and social and environmental impact, according to
the requirements specified in WP1. Also WP6 will develop tools and strategies to monitor and maintain QoS.

- WP7 is responsible for awareness and community building across the different communities of service providers, developers, consumers, vendors, standardization and regulators. WP7 will produce community profiles, methodologies & best practices, common interfaces & functionalities.
- WP8 is responsible for the dissemination and exploitation of FI-STAR productions.
- Project management and coordination is performed in WP9 including the management of Open Call experiments and the establishment of internal shared support services.

Quality processes as implemented in FI-STAR take inspiration from usual certification best practices [7]. The main actors are (Figure 6):

- WP2 to WP5: taking the role of both suppliers and customers (users) of software produced by other WPs or external suppliers (in particular GEs provided by FIWARE)
- WP6 which behaves as a certification body, aiming to be neutral evaluation place within the project. It collects all measurements made by technical WPs (WP2-5) and supply the aggregated results to the Board of Project Leaders (BPL) for review and validation
- The FI-STAR BPL acts upon received results

![Figure 6: FI-STAR quality assessment labelling model](image)

Exact participation of workpackages from WP2 to WP5 depends on the considered subject to be evaluated, the dimension analysed, the selection being made based on stakeholders interests (Table 1).

Decision has been made to focus on production originated from the FI-PPP and thus not to include software and devices provisioned from third parties.

External to FI-STAR quality evaluation will include the generic enablers, for their reusability (see §3.4.1), and the Quality of Experience as perceived from the FI-STAR developers. In case any testbed external to FI-STAR would be used, its quality of service would be evaluated.
All of the specific enablers, platforms and applications produced within FI-STAR will be evaluated. The evaluation of coverage of implemented requirements will be done for both specific enablers and applications through verification techniques (see §3.4.1). In addition specific enablers will be tested against their interoperability (when used together within applications) as well as for their reusability. Quality of Experience will be evaluated from developers’ perspective in the case of specific enablers and users’ perspective in the case of applications.

All software produced in the context of FI-STAR will be evaluated against their interoperability against health

<table>
<thead>
<tr>
<th>Table 1: focus of quality evaluation activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>FI-PPP programme</td>
</tr>
<tr>
<td>Generic enablers</td>
</tr>
<tr>
<td>Platforms (testbeds)</td>
</tr>
<tr>
<td>Specific enablers</td>
</tr>
<tr>
<td>✓ Reusability</td>
</tr>
<tr>
<td>FI-STAR project</td>
</tr>
<tr>
<td>✓ Verification</td>
</tr>
<tr>
<td>Applications</td>
</tr>
<tr>
<td>3rd parties</td>
</tr>
<tr>
<td>Infrastructure</td>
</tr>
</tbody>
</table>
3 Validation methodologies

Sections 3.1 and 3.2 describe context and overview of the FI-STAR Verification and Validation (V&V) process. The subsequent sections then specify the individual V&V instruments in detail.

3.1 Context: Overall FI-STAR software development process

WP2, WP3, and each WP4 solution runs its independent development process. Re-use of applications and GE/SE is coordinated by the respective owner of the application, GE/SE. WP1-WP6 offer WP-specific support services, standardize reports, and monitor progress and results (KPI).

WP2 and WP3 run a waterfall process with the following milestones:

- December 2013: enablers prioritized
- January 2014: scope of enabler development defined and specification of selected enablers released
- February 2014: enablers ready for black-box testing
- June 2014: enablers released

Each WP4 solution runs a two-level incremental development process:

- 2-4-weekly sprints dedicated at development themes such as a feature
- June 2014: release of alpha scope
- December 2014: release of beta scope

Coordination across FI-STAR Use Case Scenario teams and WPs is performed at multiple levels as indicated by the following subsections.

Ad-Hoc Coordination within FI-STAR Use Case Scenario Teams

Coordination within a FI-STAR Use Case Scenario team is performed upon discretion of the concerned use case owner. Any person or WP external to the use case scenario team should be involved as appropriate in a bilateral ad-hoc manner.

Coordination of WP Involvement (Heartbeat)

Coordination between FI-STAR Use Case Scenario teams with FI-STAR WPs is performed in Heartbeat phone calls. In such a meeting, development progress is reported and need for WP involvement in the Use Case Scenario development agreed.

During a heartbeat meeting, progress is acknowledged and the following actions are initiated:

- Upon finished implementation: scope and specification changes due to insights gained from development, and validation of requirements with implemented software.
- Upon finished validation: scope and specification changes due to insights gained from validation.

Releases and Major FI-STAR Milestones (Release Train)
Coordination at the FI-STAR project level is performed at the FI-STAR plenary meetings. Typical milestones include conclusion of development and initiation of trials. Physical representatives meet at these meetings and plan together the course of the overall FI-STAR project.

Transparency

Transparency of processes, plans, progress, and outcomes is achieved through a Wiki that is shared among all concerned FI-STAR partners. The Wiki also captures the FI-STAR solutions’ and project’s key performance indicators (KPI).

3.2 Quality of experience (QoE)

According to the newest working definition [8], Quality of Experience (QoE) is seen as "degree of delight or annoyance" of a user. Primarily, it is a subjective measure of user satisfaction. In fact, QoE problems are typically seen as triggers for user churn, i.e. a user may put a service aside because of bad experience, non-satisfaction, disappointment, etc. [9]. Depending on the business model, churn might entail direct or indirect loss of revenue, which explains the particular interest of service and network manufacturers, providers and operators in this matter.

QoE is the outcome of many underlying circumstances, ranging from technical parameters such as network-level Quality of Service (QoS) to content and context. Indeed, an important share of the research work on QoE focuses on quantifications of the relationships and inter-dependencies between QoE and underlying parameters, in particular QoS. For instance, a generic relationship between QoE and QoS is given in form of exponential functions [10], which typically appear in the context of waiting times. Such an exponential relationship between the QoE (here: opinion score) and the QoS (here: response time) implies that each extra amount of waiting time leads to an extra reduction of user ratings by a certain percentage (typically 10...20 % per second). The QoS "response time" can in turn be affected by other QoS parameters such as losses, link-level delays, bandwidth restrictions etc. Such relationships, which may include any number of parameters and which are not confined to QoS, are commonly denoted as QoE models.

QoE can be assessed directly by observing and/or asking users, e.g. using questionnaires, or indirectly by measuring relevant parameters within an application and/or network devices, under the condition that a valid QoE model can be used to translate QoS into QoE values. Such measurements might amongst others include usage of a feature in case a clear correlation with QoE can been derived.

In questionnaires, users are typically asked to provide

- ratings in form of (typically numerical) opinion scores (OS);
- rationales in form of comments on QoE and/or explanations of their ratings.

Quantitative ratings and qualitative rationales represent two sides of the same coin. Through the latter, users can explain their perception and subsequent rating in their own words and in detail, while the numerical rating allows for deriving quantitative relationships between cause and consequence, such as trendlines/curves and correlation factors. A nice study that demonstrates the strength of the coexistence of both ratings and rationales has been performed by HP [11].
In the FI-STAR context, different stakeholders are interested in different QoE aspects. In particular, we distinguish between the **user view** (as the addressee of FI-STAR solutions) and the **developer view** (who shall benefit from FI-PPP technology).

### 3.2.1 Generic view

This view reveals the impact of the FI-PPP technology and FI-STAR solutions on “delight or annoyance” of any stakeholder. This kind of feedback reflects whether there might be any issue with the FI-PPP technology and FI-STAR solutions that deserve further attention.

The specific items to be assessed are as follows:

- **G1. Overall satisfaction**: let the stakeholder rate and explain to which degree they (dis-) like the feature or enabler in question;
- **G2. Judgement of the feature or enabler**: in comparison to other approaches and solutions;
- **G3. Risk of churn**: ask the users directly for the risk that s/he will stop using the feature of enabler in question.

These items are contained in any of the questionnaires found in Annex A. They can be correlated (a) with the more specific ratings provided by users or developers (cf. Sections 3.2.2 and 3.2.3), and (b) with end-to-end QoS and software quality measurements (cf. Section 3.4).

When talking about ratings, we usually refer to a quantitative five-grade scale with 1 indicating the lowest/worst ranking and 5 the highest/best ranking, respectively [12]. Such numerical ratings ease numerical correlations with measurement results. The quantitative ratings are always accompanied by qualitative comments that are meant to provide additional rationales and clarifications.

### 3.2.2 User view

The above-mentioned general view can be detailed by asking a user about specific perceptions as follows:

- **U1. Perceived performance**: let the user rate and explain her perception of the response time of a feature, a typical performance measure in interactive systems. Ratings and explanations can then be correlated to measured response times as obtained from functional testing, application logs and/or instrumented interfaces that report time stamps (cf. Section 3.3.1).
- **U2. Perceived reliability**: ask the user about whether she has perceived any errors when operating the service, and about any comment on that. This user feedback can be correlated to results from functional testing and application logs (cf. Section 3.3.2).
- **U3. Perceived availability**: ask the user about whether she found the service to be non-functional, and about any comment on that. This user feedback can be correlated to usage logs on the terminal (revealing usage white-spots), watchdog measurements on the backend side (revealing unavailability) as
well as error notifications (e.g. through so-called traps) provided by a Network Management Systems (NMS) (cf. Section 3.3.3).

Item U1 may be applied to alternative performance measures such as quality of an audio connection, fluidity of a video (referring to the frequency and duration of freeze events), etc. This implies that a questionnaire may have several of those items, each of them referring to different performance aspects of the feature (as far as applicable).

On the other hand, items U2 and U3 might be out-of-question in a controlled lab environment, where availability and reliability can be guaranteed.

3.2.3 Developer view

This subsection details specific items regarding the relationship that developers have towards (Generic or Specific) Enablers.

D1. **Perceived trust**: let the developer rate and explain to which degree she trusts this enabler to be able to build the desired healthcare application. This question also incorporates reliability. It can be correlated to the list of enablers that are actually used in a specific use case.

D2. **Perceived effectiveness**: let the developer rate and explain to which degree she could do more with aid of the enabler than without it. This includes licensing issues; ease of use and integration into particular code; compatibility to (desired) development platforms; configurability; etc. The result can be correlated to the relative number (or share) of features that are using that particular enabler.

D3. **Perceived performance**: let the developer rate and explain her perception of the reaction time of a service that uses at least one enabler. Correlations can be performed towards measured response times (cf. Section 3.3.1) and to the corresponding user reactions (cf. Section 3.2.2, item U1).

D4. **Perceived ease-of use and first-time experience**: let the developer rate and explain her perception of the first-time experience in terms of “understandability”, documentation, availability and accessibility of the enabler. Correlation might be performed to the estimated time saving.

D5. **Perceived efficiency and second-time experience**: This item is similar to D4, but with focus on usage of the enabler after the learning phase, i.e. efficiency, availability and accessibility of the enabler. Again, correlation might be performed to the estimated time saving.

3.2.4 QoE Evaluation Protocol

A workshop series is performed to assess the users' "degree of delight or annoyance" about features that are ready to be released (i.e. problems of conformance to requirements were removed). The outputs of the QoE workshops are used for release decisions, correction/evolution of the implementation, and KPI reporting towards EC.

The QoE evaluation workshops are performed for the features that are ready to be released. To avoid unnecessary repetition of QoE workshops, problems of conformance to requirements should have been removed.
Below is an example of content in the case of a workshop related to users’ interactions with an evaluated application.

**Table 2: QoE evaluation workshop process**

<table>
<thead>
<tr>
<th>Participants:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• QoE evaluator</td>
</tr>
<tr>
<td>• Product (subject to be tested) owner</td>
</tr>
<tr>
<td>• Users</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Preparation:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Instrument the software with usage analytics to collect time stamps of user interactions.</td>
</tr>
<tr>
<td>• Inform <a href="mailto:samuel.fricker@bth.se">samuel.fricker@bth.se</a> and <a href="mailto:markus.fiedler@bth.se">markus.fiedler@bth.se</a> about the planned workshop.</td>
</tr>
<tr>
<td>• Optional: translate the QoE Evaluation Questionnaire (below) to users' mother tongue.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Steps:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. QoE Evaluator welcomes, defines goals of QoE evaluation, and shares agenda</td>
</tr>
<tr>
<td>2. Product owner explains the feature</td>
</tr>
<tr>
<td>3. For each user and feature:</td>
</tr>
<tr>
<td>a. User uses feature twice (2x to overcome learning of the feature).</td>
</tr>
<tr>
<td>b. User answers QoE evaluation questionnaire.</td>
</tr>
<tr>
<td>4. QoE Evaluator debriefs and thanks participants</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Materials:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• QoE Evaluation Questionnaire (see Appendix A)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Outputs:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Filled-in QoE Evaluation Questionnaires</td>
</tr>
<tr>
<td>• Time-stamp logs from user interactions.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Follow-up:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Submit outputs to <a href="mailto:samuel.fricker@bth.se">samuel.fricker@bth.se</a> and <a href="mailto:markus.fiedler@bth.se">markus.fiedler@bth.se</a> for sanity check and analysis.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>For each user group (or Persona) an individual series of QoE evaluations is performed. Results from different user groups may not be combined.</td>
</tr>
</tbody>
</table>

To enable statistical analyses, a total of 30 users or more need to be involved in the evaluation of a given feature with a given user group. One user may evaluate multiple features. QoE evaluations of a given feature may be performed at multiple, different dates. There is no need to do the evaluation with all users at the same moment. To combine the results of such a sequence of workshops, the evaluated features must not have been changed.

**Evaluation Outcome**

KPI are reported on KPI > User Satisfaction.

### 3.3 Quality of Service (QoS)

Critical answers to the generic QoE questions G.1–G.3 (cf. Section 3.2.1) and in particular user-related QoE questions U.1–U.3 (cf. Section 3.2.2) in form of low ratings and critical
comments reveal end-to-end service delivery issues. Such issues might be induced by QoS problems on network level, such as excessive packet loss, excessive packet delays, limited throughput etc. [13], [14], and/or on application level, e.g. due to server overload. Such problems might also be caused by suboptimal entity behaviour, e.g. slow and/or erroneous operation of a GE or SE, respectively.

Yet, generic monitoring of QoS-related parameters that are made available by a Network Management System (NMS) does not necessarily allow for the discovery of user-related end-to-end delivery issues due to different viewing angle and scope. Therefore, any end-to-end performance issue has to be discovered in an end-to-end fashion, followed by notifications to the NMS where correlations with the network status can be performed [14], [15]. Such a “drill-down-upon-request” strategy is rather typical within Network Management (NM):

1. The degree to which the users of FI-STAR features are “delighted or annoyed”, i.e. answer to question G.1 in the list in Section 3.2.1 provides a first indication whether there is any delivery issue to be investigated. If the user is “delighted”, there is no obvious reason to dig for potential problems. On the other hand, “annoyed” reactions (low ratings and hopefully even critical comments) and a negative answer to the churn-related question G.3 motivate a closer look at potential reasons.

2. The answers to questions U.1 to U.3 (cf. Section 3.2.2) and D.3 (cf. Section 3.2.3) allow for a differentiation of the potential source(s) of user annoyance, in particular when they can be related to in-software measurements of response times (cf. Section 3.3.1), erroneous behaviour (cf. Section 3.3.2) and availability issues (cf. Section 3.3.3).

Subsequently, the reasons for high response times and incorrect behaviours need to be identified in order to quality-assure the FI-STAR services and in order to identify potential issues with GE and SE, respectively. It is expected to derive quantitative correlations between QoE and QoS issues, revealing QoS thresholds to be met in order to provide satisfactory quality. In practice, this means that the monitoring of those specific QoS parameters will provide indications of the QoE in the given context.

### 3.3.1 Instrumentation of response time measurements

These are measures for end-to-end execution speeds, as perceived by the user (or the developer). As response times exceed certain thresholds, users perceive delays (>100ms), their flows of thought might get interrupted (>1 s), they might get bored (>4 s), and they finally might lose interest (>8...11s) [16], [17].

As opposed to one-way delay measurements, response time measurements do not suffer from the need for synchronised clocks at both measurement points – in fact, there is a single measurement point, and two time stamps need to be taken: (1) the time when the request is issued; and (2) the time when the response has been observed. Translated to the user domain, this means (1) the time when a user invokes a certain feature; and (2) the time when the user receives the result (for instance a piece of information; a viewgraph; a new page; etc.). Such time stamping can be employed in a nested manner, e.g. on subprocesses whose contributions to the overall response time delivery chain need to be quantified.
Timestamps as such are made available

- Through specific function calls, implemented at specific points of interest in the code as described above;
- Through logs as written by the application or recorded through functional testing (cf. Section 3.3.1).

The “hunt” for performance issues typically starts upon indications by users about slow performance and/or observation of response times significantly exceeding one second. Reasons for long response times may be manifold:

- large amounts of data to be sent/received;
- large processing times on the back-end side;
- lossy links;
- reordering of packets;
- large one-way delays (OWD);
- low throughputs;
- etc.

Depending on the transport protocol in use, e.g. in the case of TCP, there might be strong correlations amongst these reasons. For instance, significant losses and packet reordering cause delays and thus reflect in low throughputs. On the other hand, low link bandwidths also may imply large delays or even losses. Some of those interdependencies and their corresponding effect on QoE are available.

The monitoring approach by the XiFi project [18] follows a modular structure. The tools of choice are:

- Active and passive monitoring: PerfSONAR
- Data Center (DC) monitoring: collectd, compatible with the Monitoring GE

The minimal set of exposable parameters is as follows:

- NPP level: layer-3 statistics (SNMP), link utilisation, flow observation
- NAP level: OWD, delay variation (jitter), packet loss, achievable bandwidth, traceroute
- DC: CPU usage, memory usage, capacity

Despite of some obvious cases (link utilisation \(\rightarrow 100\%\); packet loss > 10\%; unsuccessful pings; no route; CPU usage \(\rightarrow 100\%\); memory usage \(\rightarrow 100\%\)), the correlations between concrete values of above parameters and the user perception are not given per-se. From the network point of view, non-optimal behaviour is seen from

- Extraordinarily high packet loss and jitter;
- Extraordinary (variations of) ping times;
- Extraordinary variations of layer3 SNMP statistics;
- Extraordinary variations of one-way delays and achievable bandwidth.

Still, there is a need for quantifying the impact of QoS on QoE, which is achieved by simultaneously measuring and correlating QoE and “suspicious” QoS parameters. This can happen both in a production and in a lab environment, where QoS parameters are tuned such that users report issues. This way, critical thresholds on monitorable QoS parameters can be determined and then be used for specific service monitoring.
As a final remark, response times should not be confused with ping measurements as e.g. enabled by connectd, Nagios and Zabbix. Indeed, the latter are performed on layer 3, while service response time measurements take place above layer 7 of the OSI model. Furthermore, experience shows that ping packets might be treated differently from regular data packets – sometimes, they are even blocked by operators – and that ping times do not necessarily correlate positively with available throughput, meaning that shorter ping times can imply longer download times and vice versa [19]. Thus, ping times should be interpreted with care; they might provide some lower bound estimation of the service response time, which however should be validated from case to case.

3.3.2 Instrumentation of error measurements

These errors refer mainly to abnormal behaviour of software at the user and/or the backend side. The reasons for such abnormal behaviour can range from software bugs to indefinite waiting states, e.g. caused by malfunctioning network connectivity.

The corresponding instrumentation is concentrated on the application: User- and server-sided usage logs, preferably with error codes, allow for pinpointing abnormal termination of features, e.g. through exceptions. Such service access logs are currently not foreseen in collectd, Nagios and Zabbix, but corresponding extensions may be implemented.

3.3.3 Instrumentation of availability measurements

Such errors refer to the “unreachability” of the desired feature, meaning that the user cannot even invoke it. While such misbehaviour is not necessarily an issue in a controlled lab environment, it may well be perceived as major obstacle in a real life environment.

Non-functional software does not provide logs by itself. Therefore, the major sources of (quantitative) information are user reports, from which a rate of availability issues can be derived. Furthermore, usage logs might be examined for gaps (e.g. no posts available during typical usage times). On the other hand, the back-end part of the service can be monitored with help of a watchdog function, issuing requests on a regular basis and logging the success. The latter is foreseen in connectd, Nagios and Zabbix.

3.4 Software quality

3.4.1 Verification and Validation (V&V)

FI-STAR V&V follows established good practice [20], supported by advanced Model based testing approaches (see section 4.2 on Methods for Model Based formal conformance testing).

FI-STAR V&V follows the levels of how FI-STAR software is aggregated from FI-STAR, FIWARE, and application-specific code to deployed solutions.

- Users and stakeholders depend on the end-to-end software solutions that were specified with the vision documents in D1.1.
- The end-to-end software solutions were broken down into applications to be deployed onto computing nodes, for example the mobile device and the server
inside the secure hospital context, as specified with the deployment scenarios in D1.1.

- The applications were broken into custom-developed components and enablers as specified in D4.1.
- The custom-developed components are developed by WP4, while front-end and back-end enablers are developed by WP2 and WP3 respectively.

Figure 7 gives an overview of the levels of abstraction for software development (left-hand side) and software testing (right-hand side). The green arrows, labelled “Validation”, involve checking of specifications and software against the real-world contexts into which software is to be deployed. The red arrows, labelled “Verification”, involve checking of specifications and software against other specifications.

The following types of tests are performed:

- Unit tests: testing of code as it is being produced. This involves white-box testing approaches with tests that are created by analysing the source code and thereby achieving sufficient code, branch, and path coverage.
- Conformance tests: testing of software against specifications. This involves black-box testing approaches with tests that are created by analysing state-dependent input/output behaviour and partitioning comparable behaviour into equivalence classes.
- Integration tests: testing of whether components, respectively enablers, collaborate correctly. Thereby faults in the interfaces and in the interaction between the integrated components are exposed.
- Acceptance tests: testing of whether the agreement between the stakeholders and development teams have been met and whether the solutions exhibit sufficient end-to-end quality and are accepted by users.

To monitor software quality, the following KPI are measured and reported:
- Black box testing (conformance tests): coverage of critical or important functional requirements with regression test. Preferably the tests are generated with the tools described in section 4.2. The choice is left to the application development teams (WP4) which could even decide.

- Defects (acting on test results): measurement of open defects that need to be fixed and the all defects that are open. These defect counts are used to determine whether quality is good-enough for release and use of the software.

- Interoperability with enablers: number of FI-PPP Generic or FI-STAR specific enablers in use by FI-STAR applications and number of FI-STAR application features that use a given specific, respectively generic enabler. Instead of premature architectural definitions of what Specific and Generic enablers are and how they are integrated and used, the indicators are giving transparency for each respective type of enabler.

- Other indicators, such as QoE and QoS evaluation (e.g. for acceptance tests) have been defined in the respective subsections of this document.

The white box testing (unit tests) measurements are not encouraged because of the lack of generally valid cut-off criteria and insufficient return on investment. In many situations alternative methods such as code reviews are more effective. Nevertheless, choice is left to application development teams to adapt the testing strategy to the local partner’s usages and application specificities.

The following are the responsibilities:

- WP2+3+4: Unit testing
- WP6 (Elvior): Conformance testing of enablers
- WP4: Integration testing
- WP4 (in collaboration with Elvior): Conformance testing of applications and solutions
- WP5 (in collaboration with BTH): Acceptance testing of solutions

3.4.2 Reusability

The European Commission recently adopted the Technology Readiness Levels (TRL) model developed by NASA [21] as a scale to evaluate the maturity of technology advancement of the projects funded within the EU H2020 program. To consider the specificities of software components, the NASA Earth Science Data Systems (ESDS) Software Reuse Working Group (WG) proposes a set of Reuse Readiness Levels (RRLs) aimed at measure the maturity of software for reuse [22]. In its first version [23], the RRL scale follows the TRL approach and defines 9 levels of Reuse readiness. These are:

- **RRL 1 (Limited reusability; the software is not recommended for reuse):** little is provided beyond limited source code or pre-compiled, executable binaries. There is no support, contact information for developers or rights for reuse specified, the software is not extensible, and there is inadequate or no documentation.

- **RRL 2 (Initial reusability; software reuse is not practical):** some source code, documentation, and contact information are provided, but these are still very limited. Initial testing has been done, but reuse rights are still unclear. Reuse would be challenging and cost-prohibitive.
• **RRL 3 (Basic reusability; the software might be reusable by skilled users at substantial effort, cost, and risk):** software has some modularity and standards compliance, some support is provided by developers, and detailed installation instructions are available, but rights are unspecified. An expert may be able to reuse the software, but general users would not.

• **RRL 4 (Reuse is possible; the software might be reused by most users with some effort, cost, and risk):** Software and documentation are complete and understandable. Software has been demonstrated in a lab on one or more specific platforms, infrequent patches are available, and intellectual property issues would need to be negotiated. Reuse is possible, but may be difficult.

• **RRL 5 (Reuse is practical the software could be reused by most users with reasonable cost and risk):** software is moderately portable, modular, extendable, and configurable, has low-fidelity standards compliance, a user manual, and has been tested in a lab. A user community exists, but may be a small community of experts. Developers may be contacted to request limited rights for reuse.

• **RRL 6 (Software is reusable; the software can be reused by most users although there may be some cost and risk):** software has been designed for extensibility, modularity, and portability, but software and documentation may still have limited applicability. Tutorials are available, and the software has been demonstrated in a relevant context. Developers may be contacted to obtain formal statements on restricted rights or to negotiate additional rights.

• **RRL 7 (Software is highly reusable; the software can be reused by most users with minimum cost and risk):** software is highly portable and modular, has high-fidelity standards compliance, provides auto-build installation, and has been tested in a relevant context. Support is developer-organized, and an interface guide is available. Software and documentation are applicable for most systems. Brief statements are available describing limited rights for reuse and developers may be contacted to negotiate additional rights.

• **RRL 8 (Demonstrated local reusability; the software has been reused by multiple users):** software has been shown to be extensible, and has been qualified through test and demonstration. An extension guide and organization-provided support are available. Brief statements are available describing unrestricted rights for reuse and developers may be contacted to obtain formal rights statements.

• **RRL 9 (Demonstrated extensive reusability; the software is being reused by many classes of users over a wide range of systems):** software is fully portable and modular, with all appropriate documentation and standards compliance, encapsulated packaging, a GUI installer, and a large support community that provides patches. Software has been tested and validated through successful use of application output. Multiple statements describing unrestricted rights for reuse and the recommended citation are embedded into the product.

The positioning over the scale is done by analysing the positioning within 9 topic areas being:

• **Documentation:** information that describes the software asset and how to use it.

• **Extensibility:** the ability of the asset to be grown beyond its current context.
• **Intellectual Property**: the legal rights for obtaining, using, modifying and distributing the asset.
• **Modularity**: the degree of segregation and containment of an asset or components of an asset.
• **Packaging**: the methodology and technology for assembling and encapsulating the components of a software asset.
• **Portability**: the independence of an asset from platform-specific technologies.
• **Standards Compliance**: the adherence of an asset to accepted technology definitions.
• **Support**: the amount and type of assistance available to users of the asset.
• **Verification and Testing**: the degree to which the functionality and applicability of the asset has been demonstrated.
Table 3: Reuse Readiness Level (RRL) Topic Area Levels Summary (based on [23])

<table>
<thead>
<tr>
<th>RRL Level</th>
<th>Description</th>
<th>Documentation</th>
<th>Extensibility</th>
<th>Intellectual Property Issues</th>
<th>Modularity</th>
<th>Packaging</th>
<th>Portability</th>
<th>Standards Compliance</th>
<th>Support</th>
<th>Verification and Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>RRL 1 - Limited reusability; the software is not recommended for reuse.</td>
<td></td>
<td>Little or no internal or external documentation available</td>
<td>No ability to extend or modify program behaviour</td>
<td>Product developers have been identified, but no rights have been determined.</td>
<td>Not designed with modularity</td>
<td>Software or executable available only, no packaging</td>
<td>The software is not portable</td>
<td>No standards compliance</td>
<td>No support available</td>
<td>No testing performed</td>
</tr>
<tr>
<td>RRL 2 - Initial reusability; software reuse is not practical.</td>
<td></td>
<td>Partially to fully commented source code available</td>
<td>Very difficult to extend the software system, even for application contexts similar to the original application domain</td>
<td>Developers are discussing rights that comply with their organizational policies.</td>
<td></td>
<td></td>
<td>Some parts of the software may be portable</td>
<td>No standards compliance beyond best practices</td>
<td>Minimal support available</td>
<td>Software application formulated and unit testing performed</td>
</tr>
<tr>
<td>RRL 3 - Basic reusability; the software might be reusable by skilled users at substantial effort, cost, and risk.</td>
<td></td>
<td>Basic external documentation for sophisticated users available</td>
<td>Extending the software is difficult, even for application contexts similar to the original application domain</td>
<td>Rights agreements have been proposed to developers.</td>
<td>Modularity at major system or subsystem level only</td>
<td>Detailed installation instructions available</td>
<td>The software is only portable with significant costs</td>
<td>Some compliance with local standards and best practices</td>
<td>Some support available</td>
<td>Testing includes testing for error conditions and proof of handling of unknown input</td>
</tr>
<tr>
<td>RRL 4 - Reuse is possible; the software might be reused by most users with some effort, cost, and risk.</td>
<td></td>
<td>Some extensibility is possible through configuration changes and/or moderate software modification</td>
<td>Developers have negotiated on rights agreements.</td>
<td></td>
<td></td>
<td></td>
<td>The software may be portable at a reasonable cost</td>
<td>Standards compliance, but incomplete and untested</td>
<td>Moderate systematic support is available</td>
<td>Software application demonstrated in a laboratory context</td>
</tr>
<tr>
<td>RRL 5 - Reuse is practical; the software could be reused by most users with reasonable cost and risk.</td>
<td></td>
<td>User manual available</td>
<td>Consideration for future extensibility designed into the system for a moderate range of application contexts; extensibility approach defined and at least partially documented</td>
<td>Agreement on ownership, limited reuse rights, and recommended citation.</td>
<td>Partial segregation of generic and specific functionality</td>
<td>Software is easily configurable for different contexts</td>
<td>The software is moderately portable</td>
<td>Standards compliance with some testing</td>
<td>Support provided by an informal user community</td>
<td>Software application tested and validated in a laboratory context</td>
</tr>
<tr>
<td>RRL 6 - Software</td>
<td></td>
<td>Tutorials available</td>
<td>Designed to allow</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Documentation | Extensibility | Intellectual Property Issues | Modularity | Packaging | Portability | Standards Compliance | Support | Verification and Testing |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RRL 7 - Software is highly reusable; the software can be reused by most users with minimum cost and risk.</td>
<td>Documentation</td>
<td>Extensibility</td>
<td>Intellectual Property Issues</td>
<td>Modularity</td>
<td>Packaging</td>
<td>Portability</td>
<td>Standards Compliance</td>
<td>Support</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Interface guide available</td>
<td>Demonstrated to be extensible by an external development team in a similar context</td>
<td>extensibility across a moderate to broad range of application contexts, provides many points of extensibility, and a thorough and detailed extensibility plan exists</td>
<td>recommended citation, and rights statements have been drafted.</td>
<td></td>
<td></td>
<td>software is portable</td>
<td>standards compliance with proprietary standards</td>
<td>available</td>
</tr>
</tbody>
</table>

| RRL 8 - Demonstrated local reusability; the software has been reused by multiple users. | Documentation on design, customization, testing, use, and reuse is available | Demonstrated extensibility in multiple scenarios, provides specific documentation and features to build extensions which are used across a range of domains by multiple user groups | Statements describing unrestricted rights, recommended citation, and developers embedded into product. | | | | | Organized/defined support by developer available | Software application tested and validated in a relevant context |

| RRL 9 - Demonstrated extensive reusability; the software is being reused by many classes of users over a wide range of systems. | Demonstration on design, customization, testing, use, and reuse is available | Demonstrated extensibility in multiple scenarios, provides specific documentation and features to build extensions which are used across a range of domains by multiple user groups | All functions and data encapsulated into objects or accessible through web service interfaces | | | | | Large user community with well-defined support available | Actual software application tested and validated through successful use of application output |
3.5 eHealth

3.5.1 E-Health Impact

While there has been an extensive amount of literature on e-health, but there exist demand for more solid methods of evaluating the impact of e-health. To respond to this demand, the European Commission initiated MethoTelemed study project which resulted in MAST framework.

MAST framework is based on the well-known EUnetHTA generic health technology assessment framework, while it has been extended by extensive literatures review and contribution of experts at EU level.

MAST consisted of three elements, where the first one is about preceding considerations, the second one covers seven different multi-disciplinary aspects and the third one considers transferability assessment.

The seven aspects of the second elements are as follows:

1) Health problem and characteristics of the application: Description of the health problem of the patients expected to use the telemedicine application and the application being assessed incl. description of the current use.

2) Safety: Identification and assessment of harms

3) Clinical effectiveness: Effects on the patient health

4) Patient perspectives: Acceptance of the technology, and perception of the patient or the next of kins about that.

5) Economic aspects: Comparison of the solution application with relevant alternatives in different economic aspects, including costs or expenditures-revenues business case for the healthcare institutions.

6) Organizational aspects: Resources to be recruited within an organization, and the organizational consequences expected from the use.

7) Socio-cultural, ethical, and legal aspects: The socio-cultural contexts of the patient related to the application period of the solution. The ethical challenges and questions that are raised by the applying or not applying the solution.

Also the three aspects of the third elements are cross-border, scalability, and generalizability.

It is important to note that MAST is a suggestive framework, where each project needs to customize and contextualize it, based on its own specific requirements. In the sake of this matter, we (WP6) mapped MAST with the FI-STAR value cases in vision documents (D1.1). This mapping happened through creating ontology structures in OWL language, both for FI-STAR value cases and MAST and then mapping them. This resulted in 10 aspects for evaluation for e-health impact (besides the clinical effectiveness). The list of these aspects is as follows:
1. Accessibility
2. Adhereability
3. Affordability
4. Authenticity
5. Availability
6. Efficiency
7. Effectiveness
8. Empowerment
9. Safety
10. Trustability

These quality aspects map to MAST domains as in the following table:

<table>
<thead>
<tr>
<th>FI-STAR quality aspects</th>
<th>MAST domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>QA.0 Satisfaction</td>
<td>MD.1 Health problem</td>
</tr>
<tr>
<td>QA.1 Accessibility</td>
<td>✓ MD.2 Safety</td>
</tr>
<tr>
<td>QA.2 Adherability</td>
<td>✓ MD.3 Clinical effectiveness</td>
</tr>
<tr>
<td>QA.3 Affordability</td>
<td>✓ MD.4 Patient perspectives</td>
</tr>
<tr>
<td>QA.4 Authenticity</td>
<td>✓ MD.5 Economic aspects</td>
</tr>
<tr>
<td>QA.5 Availability</td>
<td>✓ MD.6 Organisational aspects</td>
</tr>
<tr>
<td>QA.6 Efficiency</td>
<td>✓ MD.7 Socio-cultural, ethical</td>
</tr>
<tr>
<td>QA.7 Non-Clinical Effectiveness</td>
<td>✓</td>
</tr>
<tr>
<td>QA.8 Empowerment</td>
<td>✓</td>
</tr>
<tr>
<td>QA.9 Safety</td>
<td>✓</td>
</tr>
<tr>
<td>QA.10 Trustability</td>
<td>✓</td>
</tr>
<tr>
<td>MD.3 Clinical Effectiveness</td>
<td>✓</td>
</tr>
</tbody>
</table>

Also note that MD.3 (clinical effectiveness) is separated from other indicators and was not categorized as a quality attribute. (Clinical effectiveness will be the responsibility of each use case to analyse)
In the alignment process between WP6 and WP5, the total list of KPIs proposed by WP5 were submitted to a prioritization process by each use-case group, where each were asked to score KPIs as “critical”, “important”, “nice to have”. This resulted in an accumulation of these scores, with the scales weighted 9, 3, and 1 correspondingly. As those KPIs had previously been mapped with the above mentioned QA (and one MD) indicators, it resulted in prioritization of corresponding quality aspect indicators (QAs). The final result is reflected in the section 5.4.2, where more details are given about the construct of each QAs and if the way some indicators are made up as composite from several subgroups.

The measurement will be conducted through a questionnaire in a web based survey with questions that measure and build specific indicator(s).

The questionnaire will be validated in its core (English) version, but will have to be translated and validated for each language used in the use cases. This will be the responsibility of each use case. Both patients and medical personnel are targets of the questionnaire.

The measurement of e-health performance would be applied to those applications which have been officially released and delivered to their specific use-case. Enough time should be considered to ensure the supposed impacts are in effect.

### 3.5.1.1 Ethical and Social

As described above the second element of MAST covers seven different multi-disciplinary aspects, where socio-cultural and ethical aspects are one. This is an important area to cover and has been broken out of the quality indicators above to constitute a standalone set of KPIs.

The inclusion of ethical-social and cultural considerations in formal Health Technology Assessment (HTA) approaches has been pointed out by several authors [24], [25], [26] several aspects are indicated as central in HTA. Building on this approach, FI-STAR evaluation procedure will include these aspects (This inclusion is in alignment with the focus on Socio-Cultural ethical and legal aspect indicated in the MAST assessment framework).

The initial set of KPIs in this area (as described in 5.4.1) will be measured in the same web based questionnaire as the Health Impact KPIs described in 3.5.1. It is also desired later to conduct interview studies with a limited number of use cases to get a richer understanding of these often complex analyzes. Besides questions about the adherence of these applications to the classical bioethical principles of beneficence/non-maleficence and the respect of the dignity of the patient, several questions will be asked during interviews with several participants in the case studies (developers, clinical professionals, patients, hospital managers, informal caregivers). In analysis of the responses will be based on qualitative methods of discourse analysis.
3.5.2 Privacy Impact Assessment as a quality indicator

3.5.2.1 Background

This section will present Privacy Impact Assessment (‘PIA’) as a tool for assessing the quality of the FI-STAR eHealth solutions under the current EU data protection legislation and elucidate the future of PIA under the ongoing data protection reform.

Compliance with data protection has become an important quality indicator of the IT-based products and services. Consumers favoured organizations respectful of privacy in 2013 as much as they preferred ‘green’ businesses in the 1990s [27]. Some national legislation, e.g. of Schleswig-Holstein, Germany, prescribes for the public bodies, when procuring IT-based products and services, to give preference to the ones that have been certified for data protection compliance [28]. Considering sensitivity of health-related personal data, data protection compliance is a key indicator of quality of eHealth solutions. PIA provides a tool to both ensure and ascertain that the product, service or process at hand does not present or effectively mitigates data privacy risks.

3.5.2.2 Current PIA frameworks

Currently, there is no general EU legal requirement to conduct a PIA\(^3\). There is no legislative definition of a PIA either. Scholarly literature defines PIA as “a methodology for assessing the impacts on privacy of a project, policy, programme, service, product or other initiative and, in consultation with stakeholders, for taking remedial actions as necessary in order to avoid or minimise negative impacts” [29]. A PIA is a process “which should begin at the earliest possible stages, when there are still opportunities to influence the outcome of a project” and last throughout the entire lifecycle of the IT-based product or service [29]. Importantly, a PIA is not a single-shot exercise, but an ongoing process from the time when a plan to process personal data is conceived throughout the entire processing. The PIA process incorporates a feedback loop, allowing to adjust both the IT-based product and the providing organisation’s internal procedures depending on the PIA results. When a change to the data processing operation under the PIA is made, the PIA process should be repeated to account for changes and ensure continuous compliance with the data protection requirements.

Although there is no such a general legal obligation on the EU level, conducting a PIA, preferably, on an early stage of designing a system, brings a number of benefits:

- PIA is an early warning system that alerts about data privacy risks associated with a project and allows to accounts for them (e.g. build in safeguards or call back (a part of) the project altogether when the risks cannot be sufficiently mitigated);
- PIA enables to demonstrate compliance with data protection legislation. The resulting PIA report can provide evidence that the organisation took appropriate steps to prevent the occurrence, e.g. data security breach. A well-executed PIA may be used to mitigate or even exclude civil liability under particular circumstances [30].

---

\(^3\) Although Article 20 of the General Data Protection Directive on prior checking when data processing presents specific risks is considered a predecessor to PIA.
- PIA informs decision-makers within an organisation about privacy issues, exposes internal communication failures and challenges hidden assumptions about the project.
- PIA can aid in gaining public’s trust and confidence in a project, technology or service, including trust of the customers.
- PIA educates organisation’s employees and partners about the organisation’s respect to privacy and similar expectations towards employees and partners, as well as educates about privacy and privacy problems.
- An industry or organisation employing proper PIA may avoid (more burdensome) regulatory interference.\(^4\)

So far, two PIA frameworks have been submitted to Article 29 Working Party for endorsement: the 2011 PIA Framework for RFID Applications\(^5\) has been endorsed,\(^6\) and the Data Protection Impact Assessment Template for Smart Grid and Smart Metering Systems (‘DPIA Template’) has been denied endorsement [31].

### 3.5.2.2.1 PIA for RFID technologies

The RFID industry submitted an initial draft framework and, after Article 29 Working Party feedback,\(^7\) a revised PIA Framework, which received the Article 29 Working Party endorsement. Therefore, the RFID PIA Framework can be used as a model to structure PIA efforts in other sectors, with the necessary adjustments for the contexts of a given sector, e.g. healthcare.

The RFID PIA Framework establishes a PIA process of **two phases**: Initial analysis and risk assessment. An *initial analysis* phase, based on an RFID application classification and using a decision tree in Figure 1,\(^8\) allows to determine if a PIA is needed, or not, and choose a “Full Scale PIA” or a “Small Scale PIA”, varying in scope and the level of detail. For instance, applications that process personal data or use RFID tags that contain personal data require a Full Scale PIA\(^8\).

\(^{4}\) Wright 2012, p. 55


\(^{8}\) RFID PIA Framework, p. 7
The risk-assessment phase implies (1) identifying privacy risks caused by an RFID application, and (2) plan and document organisational and technical measures to mitigate those risks. The benefits of conducting the risk assessment on an early stage of system-building include saving time and costs and incorporating data protection requirements in the architecture of the system (‘privacy by design’).\(^9\) The risk-assessment phase is executed in four steps:

**Step 1:** Characterisation of an RFID application, including a comprehensive description of the application, its environment and system boundaries, interfaces with other systems, personal data flows, operation and strategic environment, e.g. stakeholders involved in information collection, the system’s mission, etc.\(^10\)

**Step 2:** Identification of risks, meaning mapping ‘conditions that may or compromise personal data’, using Data protection directive as a guide to identify privacy targets to be protected. Annexes II and III to the RFID PIA Framework contain a list of 9 privacy targets and potential privacy risks respectively. In addition, the RFID operator should consider the significance and likelihood of privacy risks occurring, deriving from both likely uses and misuses of the application, as well as the magnitude of the impact if such risks occur\(^10\).

**Step 3:** Identification and recommendation of controls, aiming at the analysis of measures that have been taken or will be taken to mitigate or eliminate the privacy risks identified in Step 2: technical measures, implemented into the application’s architecture (‘privacy by design’) like default settings, encryption, authentication, etc.; non-technical

---

\(^9\) RFID PIA Framework, pp. 7-8

\(^10\) RFID PIA Framework, p. 9
measures include management and operational procedures (some examples of ‘controls’ are given in Annex IV to the RFID PIA Framework).

Step 4: Documentation of resolution and residual risks, meaning documentation of each PIA step as well as of the final resolution concerning the RFID application at hand (approved, with relevant risks identified and addressed and no significant residual risks remaining, or not approved in its current state, requiring corrective action). Step 4 results in a PIA Report.

Both stages and their results must be documented and made available to the data protection authority in a PIA Report.

To support the execution of the PIA process, the RFID PIA Framework established a number of internal procedures, like scheduling and review of PIA, documentation, identifying triggers for a PIA revision, and stakeholder consultations.

3.5.2.2.2 PIA Methodology

Based on the Article 29 Working Party approval of the RFID PIA framework and the feedback on the rejected smart grid PIA framework, it is possible to discern general PIA methodology likely to be endorsed by the Article 29 Working Party.

Risk assessment methodology

- A PIA should be based on a risk-management approach. Hence, a PIA framework should include a risk assessment stage as a key component, also to enable evaluation of the respective risk-minimizing measures;
- A PIA should be industry-specific and not generic, both in identifying the risks and the mitigating measures;
- When identifying the risks, risks should not be confused with threats, where risks are ‘the potential that a given threat will exploit vulnerabilities of an asset or group of assets and thereby cause harm’ and threats refer to ‘the ability to exploit vulnerabilities.’
- Identifying privacy targets may help channel the PIA and compliance efforts in general. The RFIS PIA Framework identifies 9 privacy targets, based on the General data protection directive 95/46/EC. These 9 targets can be used as a model and changed to accommodate a specific context of the technology subject to PIA: (1) safeguarding quality of personal data; (2) legitimacy of data processing; (3) legitimacy

---

11 RFID PIA Framework, p. 10
12 RFID PIA Framework, p. 5
14 WP205, p. 8
15 WP205, p. 7
of processing special categories of personal data; (4) compliance with the data subject’s right to be informed; (5) compliance with the data subject’s right of access to data, correct and erase data; (6) compliance with the data subject’s right to object; (7) safeguarding confidentiality and security of processing; (8) compliance with notification requirements; (9) compliance with data retention requirements;¹⁷

- A PIA should directly address: the potential impact on a data subject (a patient or other technology user) and the privacy and data protection targets. Addressing the targets alone is not a sufficient element or a risk-based approach¹⁵. The WP29 endorsed RFID PIA Framework could be used as a model of a comprehensive PIA framework. It provides guidance how to describe the technology subject of evaluation (Annex I); privacy targets based on the Data protection directive 95/46/EC (Annex II); possible privacy risks in the area of RFID (Annex III); and a list of examples of RFID application controls and mitigating measures, both technical and organisational (Annex IV).

- A PIA should give specific guidance on how to calculate and prioritise risks, choose appropriate ‘controls’ (risk mitigating measures) and assess the residual risks. The guidance should be full and hence sufficient on its own for the implementing organizations to use, without the need to refer to external documents¹⁴;

- The identified risks should be directly matched to the mitigating measures, like in the information security standard ISO/IEC 27002:2005¹⁵;

- A risk assessment approach can build on the methodology of various national and international standards, like information security management standards (e.g. ISO/IEC 27005 [32]), and recommendations of the European Network and Information Security Agency (ENISA);¹⁸

- When assessing the risks, a special attention should be paid to what may or may not be considered personal data and hence, if data processing takes place. If a unique identifier is associated to a person, it is personal data even though it does not reveal that person’s social identity.¹⁹ For instance, if an RFID tag with a unique ID is to be carried by a person, the tag should be considered personal data;²⁰

- Identifying whether or not special categories of personal data are to be processed, and the uses of such data should be part of the risk assessment;²¹

- In identifying the risks, it is important to fully consider all risks: both intended and unintended or unauthorized uses and misuses of technology;²²

---

¹⁷ RFID PIA Framework, Annex II.
¹⁸ WP175, p. 7
²⁰ WP180, p. 5
²¹ WP175, p. 10
²² WP180, p. 5; e.g. unauthorized monitoring of RFID tags (WP175, p. 9)
• When special categories of personal data are concerned, like health data, a PIA framework should provide guidelines on how it can be processed ‘lawfully and securely’;21
• A PIA should provide guidance to determine who bears various data processing and data protection responsibilities, e.g. by means of mapping relevant actors in a given sector and helping to identify who acts as a controller or processor.23

Internal procedures
• A PIA should include a stakeholder consultation phase, involving consulting interested parties (groups, unions, associations, etc.) that can be affected. This stage should result in suggestions and improvements of both a PIA and the technology at hand;20,21
• In addition to drawing up a PIA Report and making it available to a competent authority, a concise and easy to understand information policy should be published including a summary if the PIA;24
• Each PIA framework will likely require adjustment through experience and stakeholder feedback;24
• A PIA methodology should suggest the most appropriate time for conducting a PIA in order to account for the privacy risks on the stage of designing a system to truly implement the principle of privacy by design;21

Exercise of a PIA
Drawing of the RFID PIA Framework, a PIA should include at least the phase of risk assessment where the risks specific to the technology / application are identified and risk-mitigating measures are matched to the risks. As an option, the risk assessment can be done in the four steps adopted in the RFID PIA Framework (above).

3.5.2.3 Future Data protection impact assessment
At the moment, the EU data protection framework, including its approach to Privacy Impact Assessment, is going through a reform process. In January 2012 the Commission proposed a new Data Protection Regulation (‘Commission Proposal’) [33], and in October 2013 the Committee on Civil Liberties, Justice and Home Affairs (‘LIBE Committee’) adopted its compromise text amending the Proposal. The latest reform document is the Report of the Committee on Civil Liberties, Justice and Home Affairs on the reform proposal of 21 November 2013 containing the amendments introduced by the Parliament and their explanation (‘LIBE Report’) [33]. This section’s analysis is based on the LIBE Report. Although the Council has postponed its decision on the reform till 2015, it is likely that Data Protection Impact Assessment (the term used instead of ‘Privacy Impact Assessment’) will be an important aspect of compliance with future European data protection law.

23 WP205, p. 8
24 WP180, p. 6
The LIBE Report, consistent with the Article 29 Working Party opinions on the PIA for RFID applications and smart grid, has adopted the **risk-based approach** to the Data Protection Impact Assessment (or ‘DPIA’, the term used instead of PIA). The most important change (should the LIBE amendments make it to the final text) will be that the DPIA will be made **mandatory** if certain triggers provided by law occur (Article 33 of the proposed Regulation as amended by the LIBE Report) and the risk assessment is always mandatory.

Similar to the RFID PIA Framework, and in line with the Working Party’s recommendations, the DPIA in the LIBE Report has an in-built **feedback loop** to adjust the data processing practices / technology and the DPIA processes depending of the DPIA’s outcomes. The difference is that the DPIA is only one part of that loop labelled the ‘Lifecycle Data Protection Management’ – a process of managing personal data from its collection to deletion (Recital 61).

The Lifecycle Data Protection Management is executed in four stages:

**Stage 1**: Risk analysis of intended data processing, aiming to establish the potential impact on the rights and freedoms of the data subjects, and if the intended processing is likely to present specific risks (Article 32a);

**Stage 2**: depending on the results of the risk analysis (Article 32a (3)), a controller or, where appropriate, a processor:

- designates a data protection officer; and/or
- consults the data protection officer; and/or
- carry out DPIA (Article 33).

The focus of the DPIA is the entire lifecycle of personal data from collection to deletion. The DPIA under the LIBE amendments contain, among others, a comprehensive description and purposes of the intended data processing; assessment of its necessity and proportionality; description of the measures to mitigate the risks, with due regard to the context of data processing, etc. The DPIA is followed by a periodic compliance review aiming at demonstrating compliance with the Regulation (Article 33a). Drawing on the results of the review, recommendations should be made either by the data protection officer or the national data protection authority on how to adjust data processing to achieve full compliance. Once the data processing changes compared to the processing subject to initial risk analysis, a new risk analysis under Article 32a has to be performed.

### 3.5.3 Health interoperability

To evaluate the interoperability of produced software, the project will rely on good practices and tools set by the Continua Alliance [34] as well as IHE Europe [35]. In particular, enablers claiming the applications of IHE profiles will be encouraged to participation a Connectathon event [36]. The percentage of enablers complying to the used profiles will be monitored.
4 Processes and methods for software quality

4.1 Overall process for quality monitoring

Quality monitoring takes a crucial role in the determination of success and impact of FI-PPP for healthcare. This section describes how WP6 defines and facilitates quality assessment and how the FI-STAR technical partners (WP2-5 and the use case scenario teams) implement the quality monitoring framework.

4.1.1 Context

FI-STAR development activities have the characteristics of large-scale distributed software development. Such development needs to address key concerns such as software architecture and its alignment with the development organization, elicitation and communication of requirements, sharing and adoption of methods and tools, and orchestration of project activities [37].

FI-STAR has aligned the development teams with the FI-STAR software architecture as follows (Figure 9): WP2 and WP3 are responsible for the specification and construction of the FI-STAR platform and thus facilitate adoption of FI-PPP technology. WP4 orchestrates FI-STAR application development, where one or two use case scenario teams are responsible for one application. WP1 facilitates the collection of requirements and WP5 the deployment and use of the applications in the healthcare context, and WP6 quality assurance and monitoring. The requirements, deployment, and quality assurance and monitoring work is again implemented by the respective use case scenario teams.

Figure 9: Usage relationships between healthcare ecosystem, FI-STAR applications, and FI-STAR/FI-PPP platforms. Each enabler, application, and use scenario has an owning development, respectively experimentation team.

The healthcare context affects the software and process requirements as trust in the applications and compliance with standards is important (reported in deliverable D1.2 [7]). IEC 62304 has been recommended by the European Commission as a
framework for the development of medical device-related software. It establishes safety and effectiveness of such software by enabling the sellers to demonstrate that the use of the software fulfils healthcare and solution-specific requirements and does not cause unacceptable risks.

According to the roadmaps defined in the D1.1 Vision Documents [38], the software safety classification of the FI-STAR use case scenarios does not change during the FI-STAR project. All partners have experience in delivering software that consistently meets customer requirements and applicable regulatory requirements.

4.1.2 Quality Monitoring Principles

FI-STAR quality monitoring builds on the principles of IEC 62304 and ties into enabler development, application development, and experimentation activities. FI-STAR quality monitoring aims at bringing transparency into the quality, usage, and impact of FI-STAR software and of the underlying technologies. This transparency is used to enable corrective action during the FI-STAR project and determining how FI-STAR and FI-PPP impact healthcare.

IEC 62304 is implemented here for Class A software, which does not cause injury or damage to health. Affected by the IEC 62304 standard are the development of FI-STAR applications and enablers that are deployed into their respective usage context. Applications and enablers are here considered as “black boxes” and thus the refinement and allocation of requirements to internal components considered architecture decisions within the software. Note that any use case that has not demonstrated that its software disqualifies as medical device software should follow these practices.

Table 4 describes the IEC 62304 process requirements for the development of Class A applications and enablers.

**Table 4: IEC 62304 Process Requirements for Development of Class A Applications and Enablers**

<table>
<thead>
<tr>
<th>Process Area</th>
<th>Process Requirements</th>
<th>IEC 62304 ID</th>
<th>Deliverables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality Management</td>
<td>ISO 13485 or national standard/regulation-compliant quality management system.</td>
<td>4.1</td>
<td>Partner-specific</td>
</tr>
<tr>
<td>Planning</td>
<td>Up-to-date development plan, including process and deliverables.</td>
<td>5.1.1, 5.1.2</td>
<td>FI-STAR Wiki</td>
</tr>
<tr>
<td></td>
<td>Definition of deliverable requiring verification, verification plan, and deliverable acceptance criteria</td>
<td>5.1.6</td>
<td>D4.2, D6.1</td>
</tr>
<tr>
<td></td>
<td>Coordination of V&amp;V activities with development plan</td>
<td>5.1.3</td>
<td>FI-STAR Wiki</td>
</tr>
<tr>
<td>Risk management</td>
<td>ISO 14971-compliant risk management process</td>
<td>4.2</td>
<td>Partner-specific</td>
</tr>
<tr>
<td></td>
<td>Up-to-date risk analysis and management plan</td>
<td>5.1.7</td>
<td>Partner-specific</td>
</tr>
<tr>
<td>Requirements</td>
<td>Up-to-date, consistent, unambiguous, uniquely identified application/enabler requirements for functionality, quality</td>
<td>5.2.1, 5.2.2, 5.2.5, 5.2.6</td>
<td>D1.1, D2.1, D3.1</td>
</tr>
</tbody>
</table>
(quantified performance, interoperability, alarms / warnings / operator messages, security, usability, portability, and regulatory compliance), data, operation and maintenance processes, user documentation, and risk control

<table>
<thead>
<tr>
<th>Traceability</th>
<th>solution/system requirements ↔ development plan</th>
<th>5.1.3</th>
<th>FI-STAR Wiki</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>solution/system requirements ↔ application/enabler requirements</td>
<td>5.2.6</td>
<td>Product repository, D1.1</td>
</tr>
<tr>
<td></td>
<td>Solution/system requirements ↔ conformance tests</td>
<td>5.1.1</td>
<td>Product repository</td>
</tr>
</tbody>
</table>

| Implementation | Implementation of the application/enabler | 5.5.1 | Product repository |

<table>
<thead>
<tr>
<th>Configuration and release management</th>
<th>Configuration management plan.</th>
<th>5.1.1, 5.1.9</th>
<th>Product repository</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Configuration item and system configuration documentation and history. Includes configuration items (own software, software of unknown provenance SOUP, and documentation) and their version.</td>
<td>8.1.1, 8.1.2, 8.1.3, 8.3</td>
<td>Product repository</td>
</tr>
<tr>
<td></td>
<td>Documentation plan</td>
<td>5.1.8</td>
<td>Product repository</td>
</tr>
<tr>
<td></td>
<td>Release documentation</td>
<td>5.8.4</td>
<td>Product repository</td>
</tr>
</tbody>
</table>

For information purposes, Table 5 describes the IEC 62304 process requirements for maintenance of class A application and enablers. These activities are relevant once a class A application has been released. This does not apply to FI-STAR which only develop prototypes and not products.

**Table 5: IEC 62304 Process Requirements for Maintenance of Class A Applications and Enablers.**

<table>
<thead>
<tr>
<th>Process Area</th>
<th>Process Requirements</th>
<th>IEC 62304 ID</th>
<th>Deliverables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>Maintenance plan (user feedback, problem determination and resolution, and release management)</td>
<td>6.1</td>
<td>Partner-specific</td>
</tr>
<tr>
<td>Issue Management</td>
<td>User feedback (monitoring, feedback problem report documentation, evaluation of feedback into problem categories and safety impact) and inform regulators about problems in release software products.</td>
<td>6.2.1, 6.2.5, 9.3</td>
<td>Issue tracking system</td>
</tr>
<tr>
<td></td>
<td>Problem report preparation, classification (type, scope, criticality/impact), cause investigation and safety impact evaluation and documentation, change request creation, stakeholder information, and</td>
<td>5.1.1, 6.2.2, 9.1, 9.2, 9.5, 9.6, 9.7</td>
<td>Issue tracking system</td>
</tr>
</tbody>
</table>

© Fi-STAR consortium 2013
resolution to address problem reports, and archival of problem reports, resolution, and verification, problem trend analysis, and problem resolution verification.

<table>
<thead>
<tr>
<th>Change Management</th>
<th>Change management (evaluate, analyse safety impact and approve change requests) with traceability of change request, problem report, and change request approval</th>
<th>5.1.1, 6.2.3, 6.2.4, 7.4.1, 8.2.1, 8.2.4, 9.4</th>
<th>Issue tracking system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation</td>
<td>Use of an established development or maintenance process to implement changes, verify changes</td>
<td>6.3.1, 8.2.2, 8.2.3</td>
<td>Partner-specific</td>
</tr>
<tr>
<td>Release management</td>
<td>Upon problem resolution, document tests (date, tester, software version, system configuration, tools, test results, and anomalies)</td>
<td>9.8</td>
<td>Product repository</td>
</tr>
<tr>
<td></td>
<td>Manage full or incremental software releases.</td>
<td>6.3.2</td>
<td>Product repository</td>
</tr>
</tbody>
</table>

Compliance with the IEC 62304 process requirements is the responsibility of the respective development team: the WP2 and WP3 Specific Enabler development teams and the use case scenario teams. WP6 facilitates compliance by recommending processes, methods, and tools to be used. In particular, WP6:

- Coordinates access to relevant standards, such as IEC 62304, and encourages their adoption.
- Contributes to process definition through D6.1.
- Facilitates transparency of development plans and progress through the FI-STAR Wiki.
- Recommends and facilitates use of best-practice quality assurance and monitoring methods and tools.
- Collects and spreads KPI, some of which relate to the adoption of standards.

### 4.1.3 Process and Responsibilities

To facilitate compliance with IEC 62304, to trigger the right V&V activity at the right moment, and to collect and share valid data about quality, WP6 adopts the following quality monitoring process. The process is supported by a wiki and regular “heartbeat” calls. The process is performed in cooperation with the technical reference team that consists of leaders for WP1-5, WP9, and the use case teams. Sharing of current information is critical, especially in large-scale distributed development. To avoid the typical configuration management problems when documents are used, FI-STAR has adopted a wiki [39] that centralizes information and provides access to the most current version. A wiki is a lightweight content management solution that enables collaboration on the web. FI-STAR uses a wiki to capture and share the following information:

- Process, methods, and pointers to tools and standards. The wiki contains instructions from the technical work packages about the overall engineering process, thus instructs the FI-STAR partners of how to collaborate for technical
work. It provides protocols, templates, and pointers to tools to be used for implementing methods such as model-based testing, QoE and QoS evaluation, and health-impact assessment. Finally, the wiki facilitates access to standards that affect the technical work within FI-STAR.

- Plans and status. The wiki contains up-to-date plans that inform the FI-STAR partners about goals, priorities, and milestones of technical development teams. Status is reported to share information about progress. This information allows WP6 to trigger V&V activities as outlined in Table 6.
- KPI. The wiki contains definitions of the key performance indicators that are collected by WP1-5 and reported by WP6. The wiki is also intended to share the current values of the KPI.
- Minutes of meeting. The wiki contains the minutes of the heartbeat meetings.

Awareness about plans and progress and to coordinate the work of the interdependent partners, FI-STAR has adopted a regular “heartbeat” phone call. Such calls have been effectively used to massively accelerate productivity in large-scale distributed development in comparison to loose coordination [40]. FI-STAR uses the heartbeat meetings to achieve the following objectives:

- Information about new wiki contents. The technical FI-STAR partners, in particular WP6, inform about new or changed contents on the wiki.
- Presentation of updated plans and status. The technical development teams report about software assets, development plans for these assets, and development progress.
- Presentation of updated KPI. WP6 informs about KPI and the process for collecting, analysing, and reporting the KPI. The FI-STAR partners report about KPI values.

The minutes of the heartbeat calls are spread through the FI-STAR mailing lists and archived on the wiki. Ad-hoc, coordination meetings between partners and the FI-STAR plenary meetings complement the heartbeat.

The plans, status reports, and heartbeat calls are used for triggering of V&V and quality monitoring activities. Table 6 gives an overview of the triggered activities and the milestones that trigger them.

Table 6: Triggering of V&V and quality monitoring activities (facilitated by WP6 if not indicated otherwise).

<table>
<thead>
<tr>
<th>Milestones</th>
<th>V&amp;V and Quality Monitoring Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>SE Released</td>
<td>Model Based Testing of SE Conformance</td>
</tr>
<tr>
<td>SE Integrated in Application</td>
<td>Integration Testing of SE, Developer QoE Evaluation</td>
</tr>
<tr>
<td>During Implementation</td>
<td>Unit and Integration Tests (WP4)</td>
</tr>
<tr>
<td>Application Feature Implemented</td>
<td>Testing of Feature Conformance</td>
</tr>
<tr>
<td>Application Feature Tested</td>
<td>Preliminary User QoE Evaluation and QoS Evaluation</td>
</tr>
<tr>
<td>Solution Deployed</td>
<td>Complete User QoE and QoS Evaluation (during training)</td>
</tr>
<tr>
<td>Experimentation Concluded</td>
<td>Health Impact Assessment, Sustainability Evaluation</td>
</tr>
</tbody>
</table>

To facilitate implementation of quality monitoring process, the following responsibilities have been defined. Table 7 gives an overview.
Table 7: Quality Monitoring Responsibilities.

<table>
<thead>
<tr>
<th>Responsibility</th>
<th>Partner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wiki Ownership</td>
<td>WP6</td>
</tr>
<tr>
<td>Process Definitions, Templates, and Pointers to</td>
<td>WP1-6</td>
</tr>
<tr>
<td>Tools</td>
<td></td>
</tr>
<tr>
<td>Quality Assurance and Monitoring Method Definitions</td>
<td>WP6</td>
</tr>
<tr>
<td>QA&amp;M Method Implementation</td>
<td>WP2-5, Use Case Scenario Teams (assisted by WP6)</td>
</tr>
<tr>
<td>Plans and Status</td>
<td>WP2, WP3, Use Case Scenario Teams</td>
</tr>
<tr>
<td>KPI Definitions</td>
<td>WP6</td>
</tr>
<tr>
<td>KPI Values</td>
<td>WP2-5, Use Case Scenario Teams</td>
</tr>
<tr>
<td>Minutes of Meetings</td>
<td>Meeting Scribe</td>
</tr>
<tr>
<td>Heartbeat</td>
<td>WP6</td>
</tr>
</tbody>
</table>

4.2 Methods for Model Based formal conformance testing

Interoperability has been and will continue to be one of the most important features accountable for the success of the Internet from its initiation to its current state. Obviously, this aspect must be considered by any future development of the network, independently of whether a clean-slate or an evolutionary approach is chosen. Interoperability is mainly achieved through standardization of technologies aimed at allowing the exchange of data, which is essential function of every network. This includes network architectures (e.g. ISO/OSI), communication protocols (e.g. TCP/IP), programming languages, modelling notations, data encoding notations etc. Besides interoperability testing in its various forms, conformance testing is widely accepted as a mean for facilitating interoperability in the type of heterogeneous environment that the Internet represents. Conformance testing aims at ensuring that all implementations of a given standard specification obey by the rules defined by that standard. Therefore, conformance testing can rightfully be viewed as a first step towards interoperability.

4.2.1 Black-box test automation

The task of black box testing (also known as ‘behavioural testing’ or ‘functional testing’) is to verify that the System Under Test (SUT) conforms to the specifications. The SUT is viewed as a black box, which transforms input into output according to its specifications. In automated black box testing the SUT is executed against the test tool, which acts as the environment for the SUT. The test tool provides input for the SUT and examines the actual output against that expected. The scripts that are executed by the test tool can automate black box tests. Scripts can be implemented in general-purpose programming languages like Java [41], in scripting languages like Perl [42], in test-dedicated languages like TTCN-3 [43] and more.

Although script-based automation increases black box testing productivity and tests repeatability in regression tests, test engineers have experienced serious problems in traditional script-based black box testing:

- Test engineers are rarely able to create a sufficient amount of test scripts manually to achieve adequate test coverage. Manual scripting is a time-
consuming activity and involves an effort comparable to implementing the SUT itself.

- In longer development projects the amount of test scripts may increase enormously. In the systems maintenance phase, the regression tests are used to verify that the old features of the SUT are still working after the implementation is changed. The SUT changes, which affect its external interfaces, are the reason why test scripts should be modified accordingly. In the case of a considerable number of legacy test scripts, maintenance costs may prove unacceptably high.

- The high maintenance costs of test scripts is the reason why test coverage of legacy test suites decreases over time, since only subsets of existing test scripts tend to be updated.

4.2.2 Model based testing overview

![Model Based Testing Process Diagram]

**Figure 10: The Model-Based Testing Process**

Model-Based Testing (MBT) is a generic term used to denote all testing approaches whereby model-design techniques are applied to enhance testing and affiliated activities quantitatively, qualitatively or both.

As depicted in Figure 10, the MBT process mainly consists of two activities:

- Model-Based Test Design consists in using various sources of information describing the system or technology to be tested to design a so-called test model. Among the information used for describing the system, one can name the requirements on it and a specification of the system, be it informal, formal (e.g. some formal model) or a combination of both. The test model resulting from this activity is a (semi-) formal representation of the system from a testing perspective. The testing perspective implies here that certain aspects may be highlighted in this model, that are only relevant from the point of view of testing, while others that play a less important role in that context may be hidden or
encapsulated. One of the biggest challenges in MBT is finding the right balance between those various aspects in the test model to ensure that, while it would allow the derivation/generation of test cases with the highest possible level of intended coverage, the model would still remain reasonably understandable to human readers thereof.

- Test Model Transformation consists of applying test generation algorithms or known test patterns to generate/derive executable test cases from the test model designed in the previous step. It’s worth noting here that by executable test cases, one does not necessarily mean automatically executable test cases. In fact, the level of granularity for the generated test cases is highly dependent on the level of abstraction and the “completeness” of the designed test model. Therefore, automatically executable test cases or manually executable test procedures may be the output from this activity.

In the case of FI-STAR, Model-Based Testing (MBT) refers to the automation of black box testing where the test cases (test scripts) are derived, in whole or in part, from a model that describes the expected behaviour of the SUT [44]. The MBT workflow consists of the following primary steps:

- The external behaviour of the SUT is modelled according to the relevant specifications of the SUT. The model presents the correct expected behaviour of the SUT.

- Test engineer defines the test purposes (test objectives / test goals). The tests are always generated for particular test purposes that define the scope and coverage of the generated tests.

- The tests are generated automatically from the SUT model by a test generation tool using the test purposes.

- The generated tests are executed against the SUT to verify its conformity to the model.
Figure 11: MBT workflow

MBT has the following benefits compared to traditional script-based black box testing:

- Modelling of the SUT for testing purposes has a similar effect on the quality of the IUT specification as the modelling of the SUT does on design and implementation purposes - it helps detect possible inconsistencies and ambiguities in the specification before actual testing.

- Automatic test generation is time-effective and cost-effective [45]. After creating the IUT model and defining the test goals, the remaining test generation can be fully automatic.

- Automatic test generation gives better test coverage than manually created tests [46]. Tests can be created from an SUT model for many different test goals simply by defining the new test goal and generating new tests just by pressing a button. In this way, tests for different use cases can easily be generated from the same model. This is not feasible with manually created tests: implementing all of them manually is normally too costly.

- MBT improves legacy tests maintainability. After updates in the behaviour or interfaces of the SUT there is no need to go through a huge number of legacy tests to update them accordingly. Updating the model alone is enough, and all tests can then be regenerated automatically from the model.
4.2.3 Taxonomy of model-based testing

Model-based testing is a large domain. The term ‘model-based testing’ is widely used, with slightly different meanings. Surveys on different model-based testing approaches are presented in [47], [46], [44], [48].

To understand MBT landscape as a whole, the framework of model-based testing taxonomy introduced in [48] is used below.

The taxonomy of MBT includes three general classes: the model, test generation and test execution. Each class is divided into categories. The model-related categories are subject, independence, characteristics and paradigm. The test generation class is split into test selection criteria and technology, while the test execution is divided into execution options (Figure 12).

Figure 12: Overview of the taxonomy for MBT in [48]

The subject defines what is modelled. The subject can be the intended behaviour of the SUT or the possible behaviour of the environment of the SUT, or a combination of the two.

The model for SUT black box testing expresses the intended behaviour of the SUT in its interfaces as seen from the outside of the SUT (black box). This approach assumes that the SUT is controllable and observable via its interfaces. The model expresses the behaviour of the IUT as an expected reaction observed in the observable interfaces to the external stimuli received by the controllable interfaces.

This explains why the design model of system implementation cannot be used as such for black box testing purposes. Although the sources of the models for design and test purposes are the same, they model different aspects of the systems for different purposes. The first concentrates on the inner architecture and behaviour of
implementation. The second concentrates solely on external behaviour, which is outside controllable and observable.

The independence aspect reflects the source of the model. If the model is designed directly from informal requirements specifically for testing purposes by a team that is independent of IUT developers, there will be a high degree of independence between the models created for testing and development purposes, and the testing is more likely to discover significant errors. Independence between models for testing and development purposes increases the chance that during modelling for testing purposes ambiguities will be able to be detected in the SUT specifications as a side effect of the modelling. Reusing too many existing development models in creating models for testing purposes can weaken the independence of the test suite and reduce its capabilities to detect implementation errors.

The characteristics aspect describes the properties of the model that are dependent on the characteristic of the SUT. These characteristics relate to non-determinism, to the incorporation of timing issues and to the continuous or event-discrete nature of the model and the SUT.

- Deterministic models are those where the model’s output is deterministically defined by the model input and the current state. Nondeterministic models are those where the input of the model can generate many alternative outputs in the current state of the model. Non-determinism stems partially from the internal parallel processes of the SUT, timing and hardware-related asynchronous processes. Other sources of non-determinism are the higher abstraction level of the model compared to SUT implementation and the ambiguities in the specifications of the SUT.
- Timed models are used to express the real-time constraints on the model. In general, the timing correctness of reactive systems is an important issue to test.
- The model can be discrete, continuous or a mixture of both.

The paradigm aspect reflects the style of the model and the notation of creating it. Various modelling paradigms are available. In the modelling of reactive systems the most commonly used modelling paradigms are transition-based ones. These are most natural for presenting the state-based behaviour of the SUT and the reactive relationship between SUT inputs and outputs. Typically different state machine notations are used to model reactive systems. Examples of transition-based modelling notations are:

- Finite state machine (FSM) is a notation where the nodes represent the states of the system and the transitions represent the actions or operations of the system. Textual information is used to express the input-output relationship on the transitions.
- Extended finite state machine (EFSM) is FSM with variables, guard conditions and variable assignment operations attached to transitions. EFSM is a more compact state machine presentation than FSM with improved expression power.
- Statecharts (e.g. UML State Machines [49], and STATEMATE Statecharts [50]) are EFSMs with features that increase the expressive power at the next level. Parallel and hierarchical states are represented on Statecharts.

The test selection criteria aspect describes the different test selection criteria used in test generation.
• Structural model coverage is a test selection criterion that specifies the intended test coverage in terms of model structural elements such as state and transitions.
  o Control-flow oriented coverage criteria for models are derived from similar code coverage criteria like statements, decisions, loops, and path coverage [51].
  o Data-flow oriented coverage criteria attempt to cover the control flow graph elements that define or use data variables. Examples of data-flow oriented coverage criteria are all definitions – use-pairs, all definitions, and all uses [52].
  o Transition-based coverage criteria define the transitions of the model that should be visited by generated tests. Examples of transition-based coverage are all path, all transitions, and selected transitions [44].
• Data coverage is a test selection criterion that specifies the intended test input data coverage. The three most often used data coverage criteria are:
  o Boundary value criterion selects test input values at the boundaries of the input domain [53].
  o Statistical data coverage criteria select test input values that follow a certain statistical distribution [54].
  o Pair-wise testing criteria select test input values so that all pairs of input values are tested [55].
• Requirements coverage: this aims to generate tests so that all of the requirements of the SUT are tested [44].
• Explicit test case specification: this is test selection by test engineer, who explicitly defines the test objectives of the model. The notation used to express the test objectives may be the same as the notation used for the model [44]. Notations commonly used for test objectives include FSM, UML Testing Profile (UTP), regular expressions, temporal logic formulas, constraints and Markov chains [56].
• Random and stochastic criteria: a typical approach is to use a Markov chain [57] to specify the expected SUT usage profile. Another example is to use a statistical usage model in addition to the behavioural model of the SUT [58].
• Fault-based criteria: these rely on knowledge of typically occurring faults, often designed in the form of a fault model.

The test generation technology paradigm describes the different underlying technologies used in MBT methods:
• Manual/automatic: tests can be generated automatically by test generation tools or developed manually.
• Random generation: random generation of tests is done by sampling the input space of a system (monkey tests).
• Graph search algorithms: dedicated graph search algorithms include node or arc coverage algorithms such as the Chinese Postman algorithm [59], which covers each arc at least once.
• Model checking: model checking is a technology used to verify the properties of a system using reachability analysis. The general idea of test case generation with a model checker is to first specify the test case in terms of reachability properties e.g. “eventually, a certain state is reached or a certain transition fires”. A model checker then yields traces which reach the given state or which makes the
transition fire eventually. Different model checking techniques exist e.g. explicit state model checking and symbolic model checking.

- **Symbolic execution**: the idea behind symbolic execution is to run an executable model with sets of input constraints instead of single input values to generate traces [60]. To derive test cases these traces are instantiated with concrete input values.
- **Theorem proving**: this is used to check the satisfiability of formulas in the models [61]. Theorem provers can be used in test generation in a similar way to model checkers, with a theorem prover replacing model checker.
- **Online/offline test generation** defines whether the tests are generated in advance of test execution (offline test generation) or simultaneously with it (online test generation). The term ‘on-the-fly’ is often used for online test generation. Offline test generation is mostly used to generate tests for deterministic SUT, while on-the-fly generation is mostly used with nondeterministic SUT.

### 4.2.4 Overview of MBT tools

Many MBT solutions are restricted to generating the test from deterministic models only. The following model-based testing tools fall into this category: Elvior TestCast MBT [62], Conformiq Qtronic [63], Leirios Test Generator [64] (marketed as Smartesting), Agatha [65], SpecExplorer [66], NModel [67], ATG [68], MiLEST [56], and Reactis Tester [69].

Tools that provide on-the-fly test generation for the testing of nondeterministic systems include SpecExplorer [66], NModel [67], and Uppaal TRON [70].

Modelling notations used by model-based testing tools can be divided into the following classes: textual modelling languages, FSM, EFSM and Statecharts.

Textual modelling language Spec# is used in SpecExplorer and C# is used in NModel. These textual languages have the expression power that enables the modelling of almost any aspects of the system at a very detailed level. Textual modelling languages are hard for test engineers with limited programming skills to learn. They lack graphical relationships between model components.

FSMs have received extensive theoretical study by the model-based testing community. Researchers have been working on FSM test generation algorithms since the early 1960s. Several complete test generation methods have been invented for generating tests that guarantee the IUT implementation identity with the corresponding FSM model. Unfortunately, those algorithms are not typically used in industrial practice because they are too restrictive in that they make strong assumptions about the model and the SUT. FSMs lack the expression power required to model industrial-scale systems. Serious models built in FSM grow to be huge and lack readability. NModel [67], for example, supports FSM models in addition to C# model programs.

Statecharts [50] is a state machine with hierarchical and parallel states. It is a powerful state machine notation, which is widely accepted in the industry. Unfortunately, test generation from Statecharts presents many obstacles, which are hard to overcome due to the existence of parallel states. Algorithms are available for flattening the Statecharts [71], [72], but due to this flattening the traceability from generated tests back to the original model becomes difficult. Tracing generated tests back to the model
is urgently needed during test execution to identify the aspects of the model that implementation violates. Statecharts modelling notation is used with some limitations in Conformiq’s tool [63], ATG [68], and Elvior TestCast MBT [62].

EFSM has weaker expression power compared to the Statecharts. EFSM is a compact presentation of FSM, EFSM as a graphical presentation over textual modelling notations is that it is more readable and understandable, but there are still people who prefer only textual presentations. The choice of the modelling language can be considered a matter of taste. EFSM, like FSM, is a theoretically well-studied notation. There are a large number of tools and techniques for manipulating EFSMs. The availability of model checking tools for experimenting with test generation from EFSM was one of the reasons for its selection for the modelling notation in this research. EFSMs have gained wide acceptance in software modelling and are used as semantic models for specification languages such as Statecharts and UML state machines. The timed automata notation used by the Uppaal TRON is an extension of EFSM also.

Model structural coverage criteria are used for test selection, which is based on finding transition sequences that cover user-defined model elements (coverage items). Tests are generated according to the following test coverage criteria: selected states, selected transitions, all transitions, all states, and all transition pairs. The requirements coverage criterion is implemented by combining the structural coverage items. Test selection based on model structural coverage criteria is used by the following model-based testing tools: Conformiq [63], Leirios [64], Agatha [65], and Uppaal TRON [70].

4.2.5 MBT approach proposed for FI-STAR

The functional testing of the systems and system components in FI-STAR should stand on the following pillars:

- System under test (SUT) is any system or system component that interacts to the environment through its interface.
- Test system validates that the system behaves as specified on its interfaces. This is classical black-box testing setup.
- Tests are executed automatically to facilitate continuous integration (CI) of the developed systems.
- Model based testing is used for automatically generating the test scripts.
  - Functional requirements of the SUT on its interface are specified.
  - Formal behavioural model is built based on the functional requirements.
  - Out of the model the test cases are generated automatically based on the test objectives.

The application of black box testing in the project is to cover two discrete areas with two fairly separate approaches.

The first area to be covered is Generic Enablers provided by the FI-WARE and/or other FI-PPP projects as well as any Specific Enablers developed within the FI-STAR project. The idea with applying testing here is to ensure that any Generic/Specific Enablers used within the FI-STAR project conform to the specification’s set forth for them. We would like to be able to validate these for our use case teams and be able to ensure for our use case teams that they can use these enablers and be sure they will do what they are
advertised to do. In this application individual Generic/Specific enablers will act as the black box.

The other area that black box testing could be applied in this project is to the applications and solutions developed by the use case teams. Here the goal is to use the expertise brought into the project in WP6 to help the use case teams that would like to validate their own systems using such techniques. State of the art model based testing approaches would then be used to the end user interfaces to proof entire systems from end to end.

4.2.5.1 Validation of Generic/Specific Enablers

For validation of GE/SE’s we plan to primarily use traditional automated testing via TTCN-3. The criteria for these tests will be the documents specifying the intended function of the GE or SE.

![Diagram](image)

**Figure 13: From specification documents to executable TTCN-3 test cases.**

From the specification documents available from FI-WARE or within the FI-STAR project, executable test suites will be derived (Figure 13). For each request and response to and from the GE/SE a corresponding request and response will be defined in TTCN-3. Another key part of this approach is that the test cases will be able to support multiple formats. So if a GE/SE claims to be able to support requests with bodies in both XML and JSON, both can be tested by this single test suite simply by changing the encoding settings. Since the tests here are based upon the specification for the GE/SE and not the specific implementation they can also be reused to test any implementation of that GE/SE. So if one of the use cases chooses to use an implementation developed by one partner and another use case wants to use the same GE developed by another partner, the same test suite can be used for validation of both.
The test suite will be executed against the target generic enablers via adapters, primarily HTTP based adapters that will send the messages on to the generic enabler, collect the resulting response and send that back to the testing tool after decoding it into TTCN-3 format.

It is expected that just a handful of adapters and codecs will be need and can then be reused for the testing of other generic enablers. Many of these codecs and adapters are already in place.

4.2.5.2 Validation of Use Case Applications

While most GE’s are RESTful web API’s that do not lend as well to behavioural based testing, Use Case solutions are complete solutions with finite behaviours that lend much better to model based approaches to testing. We propose to apply model based testing with finite state machines in any use case that wishes to engage with us.

Model based testing with finite state machines (MBT) is a method for testing by which functional requirements for the system are formalized into a finite state machine that represent the expected behaviour of the system. Powerful testing tools are then used to automatically design and generate test cases that will test with a high degree of accuracy weather or not the system under test (SUT) in fact exhibits the behaviour defined in the model.
Figure 15: Buidling a UML test model

Someone who is familiar with the requirements and with UML works with the requirements as well as the stakeholders responsible for the requirements to come up with a UML state machine, as well as some data definitions defining the terms used on the model. This is really where most of the effort in MBT is put in as often somewhat abstract requirements now have to be quantified and turned into actual expected behaviour. However everything else in the MBT workflow is automated. So this effort while overall should be less than the effort needed to conduct more traditional methods of testing.

Model based testing also has other intrinsic benefits. The process of modelling is very much taking requirements and deriving how the system should act from them. So the process of modelling will very often bring to the surface inaccuracies, contradictions and weaknesses within the requirements. We expect that the process of modelling will also act as a check and refinement process for some of the requirements.

From the model, tests are then derived. This is accomplished by stepping through the model, where each step is a transition. The user of the MBT tools first defines an objective for test generation. This is some specific part of the model or system that the user wishes to cover in tests. The test generation engine that steps through the model in a semi random fashion. Advanced model checking software is used to check if a step taken is valid or not and each step is also evaluated as to whether it gets closer to the goal or not. This process is repeated in an iterative manner until the entire user defined test objective is covered.
The output from test generation is abstract test cases. The meaning of an abstract test case is that this test case is not yet actually executable. It is more a plan for what an executable test case would look like. At this point the user should review the result of test generation, if the resulting abstract test case is sufficient than it can be rendered into an executable form and run against the SUT. One other key thing of note here is that because the format which the abstract test case is to be executed in is not set at this point. It may be possible to use the same test case and render it into two or more executable formats. For example in a mobile application that is both in IOS and Android a single model and test case could be used and then could be generated separately for each platform.

4.2.6 Data collection and analysis

The model based and TTCN-3 testing tools provided by Elvior will create logs at execution that can be saved later analysis. This will be very helpful in tracking some of the performance statistics, some of the key performance indicators as well as more general statistics such as fail rates. If it is specifically requested Elvior may be able to develop solutions for collecting other relevant information related to key performance indicators and other V&V markers covered elsewhere in this document. Further discussion on exactly what those are and how technically a solutions for collecting them could be developed should be had.
5 Quality indicators

This section lists the quality indicators included in the FI-STAR quality assessment framework. Each quality indicator is characterized by:

- **ID**: unique identifier.
- **Indicator name**: short name of the quality indicator.
- **Description**: longer description of the indicator aimed at reducing the risk of misinterpretation.
- **Scale**: range of values (including qualitative estimates) which can be taken by the indicator.
- **Typical target**: what is the typical value to be expected for high quality system. This value is to be tailored for each subject being included in the quality assessment framework.
- **Method**: description of the method to be used to get a measure for the indicator.
- **Provider**: person, group of persons or roles in charge to provide measures for the indicator.
- **Caveats**: warnings or clarifications regarding the interpretation and use of the indicator.
- **Updates**: periodicity at which the indicator is updated.

### 5.1 Quality of experience

<table>
<thead>
<tr>
<th>ID</th>
<th>Indicator</th>
<th>Description</th>
<th>Scale</th>
<th>Typ. target</th>
<th>Method</th>
<th>Provider</th>
<th>Caveats</th>
<th>Updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>QoE-01</td>
<td>Overall MOS</td>
<td>Mean Opinion Score MOS: how the feature is perceived overall by users</td>
<td>MOS Scale 1-5</td>
<td>4</td>
<td>Questionnaire with 5-level scale filled-in by user after walkthrough of feature use</td>
<td>Product manager</td>
<td>MOS is not the same as risk for churn</td>
<td>Feature release</td>
</tr>
<tr>
<td>QoE-02</td>
<td>Quality Impression</td>
<td>How good the feature is in the eyes of the user and in comparison to alternatives</td>
<td>Exceptional, Better than alternative, Good-enough, Insufficient</td>
<td>Good-enough</td>
<td>Questionnaire</td>
<td>Product manager</td>
<td>User may not know any alternative</td>
<td>Feature release</td>
</tr>
<tr>
<td>QoE-03</td>
<td>Risk of Churn</td>
<td>Whether the user would turn to a non-FI-STAR product.</td>
<td>Yes, No</td>
<td>90% No</td>
<td>Questionnaire</td>
<td>Product manager</td>
<td>-</td>
<td>Feature release</td>
</tr>
<tr>
<td>--------</td>
<td>---------------</td>
<td>-----------------------------------------------------</td>
<td>---------</td>
<td>--------</td>
<td>---------------</td>
<td>-----------------</td>
<td>----</td>
<td>----------------</td>
</tr>
<tr>
<td>QoE-04</td>
<td>Performance MOS</td>
<td>How the users judge the response time of the feature</td>
<td>MOS Scale 1-5</td>
<td>4</td>
<td>Questionnaire with 5-level scale filled-in by user after walkthrough of feature use</td>
<td>Product manager, respectively use case or WP2/3 project leader</td>
<td>End-user and developer perspective</td>
<td>Feature release</td>
</tr>
<tr>
<td>QoE-05</td>
<td>Reliability MOS</td>
<td>How the users judge the reliability of the feature</td>
<td>MOS Scale 1-5</td>
<td>4</td>
<td>Questionnaire with 5-level scale filled-in by user after walkthrough of feature use</td>
<td>Product manager</td>
<td>End-user perspective</td>
<td>Feature release</td>
</tr>
<tr>
<td>QoE-06</td>
<td>Availability MOS</td>
<td>How the users judge the availability of the feature</td>
<td>MOS Scale 1-5</td>
<td>4</td>
<td>Questionnaire with 5-level scale filled-in by user after walkthrough of feature use</td>
<td>Product manager</td>
<td>End-user perspective</td>
<td>Feature release</td>
</tr>
<tr>
<td>QoE-07</td>
<td>Usability MOS</td>
<td>How the users judge the usability of the feature</td>
<td>MOS Scale 1-5</td>
<td>4</td>
<td>Questionnaire with 5-level scale filled-in by user after walkthrough of feature use</td>
<td>Product manager</td>
<td>End-user perspective</td>
<td>Feature release</td>
</tr>
<tr>
<td>QoE-08</td>
<td>Trust MOS</td>
<td>How the users trust the enabler for use in a healthcare context</td>
<td>MOS Scale 1-5</td>
<td>4</td>
<td>Questionnaire with 5-level scale filled-in by user after integration of enabler</td>
<td>Use case or WP2/3 project leader</td>
<td>Developer perspective</td>
<td>SE integration</td>
</tr>
<tr>
<td>QoE-09</td>
<td>Effectiveness MOS</td>
<td>How the users judge the possibilities (in terms of features that could be developed) for the use of the enabler</td>
<td>MOS Scale 1-5</td>
<td>4</td>
<td>Questionnaire with 5-level scale filled-in by user after integration of enabler</td>
<td>Use case or WP2/3 project leader</td>
<td>Developer perspective</td>
<td>SE integration</td>
</tr>
<tr>
<td>QoE-10</td>
<td>Learnability MOS</td>
<td>How the users judge the development efficiency for the first use of the enabler</td>
<td>MOS Scale 1-5</td>
<td>4</td>
<td>Questionnaire with 5-level scale filled-in by user after integration of enabler</td>
<td>Use case or WP2/3 project leader</td>
<td>Developer perspective</td>
<td>SE integration</td>
</tr>
</tbody>
</table>
### 5.2 Quality of service

<table>
<thead>
<tr>
<th>Id</th>
<th>Indicator</th>
<th>Description</th>
<th>Scale</th>
<th>Typ. target</th>
<th>Method</th>
<th>Provider</th>
<th>Caveats</th>
<th>Updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>QoS-01</td>
<td>Performance</td>
<td>Response time</td>
<td>sec</td>
<td>95% of all responses are within 3 sec</td>
<td>Timestamp logs. Same as for QoE evaluation, but during all runtime.</td>
<td>Product Manager</td>
<td></td>
<td>End of each trial</td>
</tr>
<tr>
<td>QoS-02</td>
<td>Reliability</td>
<td>Error frequency</td>
<td>%</td>
<td>99.9% of all usage should be without errors</td>
<td>Errors logs. Same as for QoE evaluation, but during all runtime.</td>
<td>Product Manager</td>
<td></td>
<td>End of each trial</td>
</tr>
<tr>
<td>QoS-03</td>
<td>Availability</td>
<td>Usage frequency</td>
<td># per time unit</td>
<td>above threshold</td>
<td>Timestamp logs. Same as for QoE evaluation, but during all runtime.</td>
<td>Product Manager</td>
<td>Assumes user intention. Threshold to be calibrated based on intended usage of the application.</td>
<td>End of each trial</td>
</tr>
<tr>
<td>QoS-04</td>
<td>Availability</td>
<td>Frequency of provided answers</td>
<td>%</td>
<td>99.9% of all requests an answer is provided</td>
<td>Watchdog for monitoring back-end services</td>
<td>Product Manager</td>
<td></td>
<td>End of each trial</td>
</tr>
</tbody>
</table>

© Fi-STAR consortium 2013  Page 67 of (94)
## 5.3 Software quality

### 5.3.1 Verification

<table>
<thead>
<tr>
<th>Id</th>
<th>Indicator</th>
<th>Description</th>
<th>Scale</th>
<th>Typ. target</th>
<th>Method</th>
<th>Provider</th>
<th>Caveats</th>
<th>Updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sof-Ver-01</td>
<td>Requirements Coverage</td>
<td>Coverage of critical or important functional requirements with regression tests</td>
<td>%</td>
<td>All released critical and important functional requirements covered</td>
<td>Traceability table, possibly maintained with test management tool</td>
<td>Test Lead of Use Case Team</td>
<td>Quality requirements are evaluated with other V&amp;V techniques. Breakdown into features maintained within use case team.</td>
<td>Feature release</td>
</tr>
<tr>
<td>Sof-Ver-02</td>
<td>Open Defects</td>
<td>Number of critical or important defects (bugs) that are open and need to be fixed</td>
<td>#</td>
<td>No known critical or important defects in release</td>
<td>Issue tracker like Jira or Redmine</td>
<td>Test Lead of Use Case Team</td>
<td>No differentiation between defects and failures. Breakdown into criticality, modules, and features maintained within use case team.</td>
<td>End of sprint</td>
</tr>
<tr>
<td>Sof-Ver-03</td>
<td>All Defects</td>
<td>Total number of known critical or important defects (bugs), including open and closed defects</td>
<td>#</td>
<td>Converge to zero when approaching a release</td>
<td>Issue tracker like Jira or Redmine</td>
<td>Test Lead of Use Case Team</td>
<td>No differentiation between defects and failures</td>
<td>End of sprint</td>
</tr>
</tbody>
</table>
### 5.3.2 Internal interoperability

<table>
<thead>
<tr>
<th>Id</th>
<th>Indicator</th>
<th>Description</th>
<th>Scale</th>
<th>Typ. target</th>
<th>Method</th>
<th>Provider</th>
<th>Caveats</th>
<th>Updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sof-Int-01</td>
<td>Number of Specific Enablers</td>
<td>Number of FI-STAR Specific Enablers</td>
<td># SE</td>
<td>&gt;= 5</td>
<td>Report</td>
<td>WP2 and WP3 leaders</td>
<td>Upon FI-STAR BPL meetings</td>
<td></td>
</tr>
<tr>
<td>Sof-Int-02</td>
<td>Number of successful uses by features</td>
<td>Number of FI-STAR application features that use the SE</td>
<td># Features</td>
<td>&gt;= 5</td>
<td>Report</td>
<td>WP4 application development team leaders</td>
<td>End of Sprint</td>
<td></td>
</tr>
</tbody>
</table>

### 5.3.3 Reusability

<table>
<thead>
<tr>
<th>Id</th>
<th>Indicator</th>
<th>Description</th>
<th>Scale</th>
<th>Typ. target</th>
<th>Method</th>
<th>Provider</th>
<th>Caveats</th>
<th>Updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sof-Reu-01</td>
<td>Documentation</td>
<td>Information that describes the software asset and how to use it</td>
<td>1-9</td>
<td>9</td>
<td>Questionnaire</td>
<td>Experimented user</td>
<td>Subjective analysis</td>
<td>Major release</td>
</tr>
<tr>
<td>Sof-Reu-02</td>
<td>Extensibility</td>
<td>The ability of the asset to be grown beyond its current context</td>
<td>1-9</td>
<td>9</td>
<td>Questionnaire</td>
<td>Experimented user</td>
<td>Subjective analysis</td>
<td>Major release</td>
</tr>
<tr>
<td>Sof-Reu-03</td>
<td>Intellectual Property</td>
<td>The legal rights for obtaining, using, modifying and distributing the asset</td>
<td>1-9</td>
<td>9</td>
<td>Questionnaire</td>
<td>Experimented user</td>
<td>Subjective analysis</td>
<td>Major release</td>
</tr>
<tr>
<td>Sof-Reu-04</td>
<td>Modularity</td>
<td>The degree of segregation and containment of an asset or components of an asset</td>
<td>1-9</td>
<td>9</td>
<td>Questionnaire</td>
<td>Experimented user</td>
<td>Subjective analysis</td>
<td>Major release</td>
</tr>
<tr>
<td>Sof-Reu-05</td>
<td>Packaging</td>
<td>The methodology and technology for assembling and encapsulating the components of a software asset</td>
<td>1-9</td>
<td>9</td>
<td>Questionnaire</td>
<td>Experimented user</td>
<td>Subjective analysis</td>
<td>Major release</td>
</tr>
<tr>
<td>Sof-Reu-06</td>
<td>Portability</td>
<td>The independence of an asset from platform-specific technologies</td>
<td>1-9</td>
<td>9</td>
<td>Questionnaire</td>
<td>Experimented user</td>
<td>Subjective analysis</td>
<td>Major release</td>
</tr>
</tbody>
</table>
## 5.4  E-health

### 5.4.1  Ethical and Social

<table>
<thead>
<tr>
<th>Id</th>
<th>Indicator</th>
<th>Description</th>
<th>Scale</th>
<th>Typ. target</th>
<th>Method</th>
<th>Provider</th>
<th>Caveats</th>
<th>Updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ehe-eth-01</td>
<td>Privacy impact assessment (PIA)</td>
<td>A privacy impact assessment process exist</td>
<td>Yes/No</td>
<td>Yes</td>
<td>Specialist evaluation</td>
<td>WP6</td>
<td>No details related to the PIA results</td>
<td>End of project</td>
</tr>
<tr>
<td>Ehe-eth-02</td>
<td>User Involvement in Design</td>
<td>If, and to what extent the patients or patient or the professionals were involved in the design of the application</td>
<td>Likert (1-5)</td>
<td>4</td>
<td>Questionnaire</td>
<td>Use Case Owner</td>
<td>End of each trial</td>
<td></td>
</tr>
<tr>
<td>Ehe-eth-03</td>
<td>User Exclusion</td>
<td>If the design of the application exclude some groups (religious, cultural, social communities, or racial minorities) (social exclusion) or if the application is in conflict with religious, social or cultural convictions/values?</td>
<td>Likert (1-5)</td>
<td>4</td>
<td>Questionnaire</td>
<td>Use Case Owner</td>
<td>End of each trial</td>
<td></td>
</tr>
<tr>
<td>Ehe-eth-04</td>
<td>Professional-Patient Relationship</td>
<td>If, and to what extent the application affects the doctor-patient relationship</td>
<td>Likert (1-5)</td>
<td>4</td>
<td>Questionnaire</td>
<td>Use Case Owner</td>
<td>End of each trial</td>
<td></td>
</tr>
</tbody>
</table>
### Ehe-eth-05 Informal Caregivers
- **Does the application have positive impact informal caregivers (close relatives or friends of the users)?**
  - Likert (1-5)
  - 4
  - Questionnaire
  - Use Case Owner
  - End of each trial

### Ehe-eth-06 Informal Caregivers
- **Does the application have positive impact informal caregivers (close relatives or friends of the users)?**
  - Likert (1-5)
  - 4
  - Questionnaire
  - Use Case Owner
  - End of each trial

### Ehe-eth-07 Social Environment Impact
- **Does the application have positive impact negatively affect the user’s relation with his/her social environment?**
  - Likert (1-5)
  - 4
  - Questionnaire
  - Use Case Owner
  - End of each trial

## 5.4.2 Health impact

<table>
<thead>
<tr>
<th>Id</th>
<th>Indicator</th>
<th>Description</th>
<th>Scale</th>
<th>Typ. target</th>
<th>Method</th>
<th>Provider</th>
<th>Caveats</th>
<th>Updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ehe-Imp-01</td>
<td>General Satisfaction</td>
<td>The degree to which the application’s healthcare output is satisfactory in its entirety.</td>
<td>Likert (1-5)</td>
<td>4</td>
<td>Questionnaire</td>
<td>Use Case Owner</td>
<td></td>
<td>End of each trial</td>
</tr>
<tr>
<td>Ehe-Imp-02</td>
<td>Accessibility</td>
<td>The degree to which the health care service delivered through the application can be easily accessed by different group of people.</td>
<td>Likert (1-5)</td>
<td>4</td>
<td>Questionnaire</td>
<td>Use Case Owner</td>
<td></td>
<td>End of each trial</td>
</tr>
<tr>
<td>Ehe-Imp-03</td>
<td>Adhereability</td>
<td>The degree to which the application helps patients to more adherence by some sort of motivators such as being a member of a community.</td>
<td>Likert (1-5)</td>
<td>4</td>
<td>Questionnaire</td>
<td>Use Case Owner</td>
<td></td>
<td>End of each trial</td>
</tr>
<tr>
<td>Ehe-Imp-04</td>
<td>Affordability</td>
<td>The degree to which the healthcare service delivered through the application is more affordable or has decreased total expenditures, specially when comparing with other alternatives.</td>
<td>Likert (1-5)</td>
<td>4</td>
<td>Questionnaire</td>
<td>Use Case Owner</td>
<td></td>
<td>End of each trial</td>
</tr>
<tr>
<td>Ehe-Imp-05</td>
<td>Authenticity</td>
<td>The degree to which the application improves the authenticity of some health related product or information.</td>
<td>Likert (1-5)</td>
<td>4</td>
<td>Questionnaire</td>
<td>Use Case Owner</td>
<td></td>
<td>End of each trial</td>
</tr>
<tr>
<td>Ehe-Imp-06</td>
<td>Availability</td>
<td>The degree to which the healthcare service, provided through the application, is available when needed.</td>
<td>Likert (1-5)</td>
<td>4</td>
<td>Questionnaire</td>
<td>Use Case Owner</td>
<td></td>
<td>End of each trial</td>
</tr>
<tr>
<td>Ehe-Imp-07</td>
<td>Efficiency</td>
<td>Likert (1-5)</td>
<td>Questionnaire</td>
<td>Use Case Owner</td>
<td>End of each trial</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------------------------------------------------</td>
<td>--------------</td>
<td>---------------</td>
<td>----------------</td>
<td>-------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The degree to which the healthcare service complexity, number of tasks, or</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>time consumption has decreased.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ehe-Imp-08</td>
<td>Non-Clinical Effectiveness</td>
<td>Likert (1-5)</td>
<td>Questionnaire</td>
<td>Use Case Owner</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The degree to which the healthcare service produces better results (e.g. no</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>clinical mistakes, improved knowledge, more readiness, and more evidences</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>for personalized treatment).</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ehe-Imp-09</td>
<td>Empowerment</td>
<td>Likert (1-5)</td>
<td>Questionnaire</td>
<td>Use Case Owner</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The degree to which the patients and medical personnel are empowered</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(e.g. by increasing knowledge about the specific patient situation or about</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>the disease in general).</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ehe-Imp-10</td>
<td>Safety</td>
<td>Likert (1-5)</td>
<td>Questionnaire</td>
<td>Use Case Owner</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The degree to which the application is considered to be safe, both for</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>patients and medical personnel, and both by being safe itself and also</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>helping to improve the safety in healthcare.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Being safe can be achieved by being no cause of disability, morbidity, or</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>mortality, avoiding misleads and confusing information, and to has no (or</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>minimal) possible harm impact even when it fails. The case of safety even</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>when the application fails, can be characterized by being cause of no harm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(or minimal), very rare happening of harm, and very short duration time of</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>harm during failure.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This indicator also corresponds to actively providing guide on minimising</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>the risks during the operation, detecting emergency situations, or detecting</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>unsafe behaviours or glitches in the healthcare process</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ehe-Imp-11</td>
<td>Trustability</td>
<td>Likert (1-5)</td>
<td>Questionnaire</td>
<td>Use Case Owner</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The degree to which the application attains trust of patients (for example</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>by ensuring privacy of their information or being non-invasive in its</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>interaction with them).</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 5.4.3 eHealth interoperability

<table>
<thead>
<tr>
<th>Id</th>
<th>Indicator</th>
<th>Description</th>
<th>Scale</th>
<th>Typ. target</th>
<th>Method</th>
<th>Provider</th>
<th>Caveats</th>
<th>Updates</th>
</tr>
</thead>
</table>

Page 72 of (94) © FI-STAR consortium 2013
5.5 Quality targets

The tables below provide for each proposed indicator, a quality target to be reached within the project.

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Subjects (to be further differentiated)</th>
</tr>
</thead>
<tbody>
<tr>
<td>QoE-01 Overall MOS</td>
<td>4 Generic enablers</td>
</tr>
<tr>
<td>QoE-02 Quality Impression</td>
<td>Good-enough</td>
</tr>
<tr>
<td>QoE-03 Risk of Churn</td>
<td>90% No</td>
</tr>
<tr>
<td>QoE-04 Performance MOS</td>
<td>4</td>
</tr>
<tr>
<td>QoE-05 Reliability MOS</td>
<td>4</td>
</tr>
<tr>
<td>QoE-06 Availability MOS</td>
<td>4</td>
</tr>
<tr>
<td>QoE-07 Usability MOS</td>
<td>4</td>
</tr>
<tr>
<td>QoE-08 Trust MOS</td>
<td>4</td>
</tr>
<tr>
<td>QoE-09 Effectiveness</td>
<td>4</td>
</tr>
<tr>
<td>Indicator</td>
<td>Subjects (to be further differentiated)</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td><strong>Id</strong></td>
<td><strong>Name</strong></td>
</tr>
<tr>
<td>MOS</td>
<td>Learnability MOS</td>
</tr>
<tr>
<td>QoE-10</td>
<td></td>
</tr>
<tr>
<td>QoE-11</td>
<td>Efficiency MOS</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>QoS-01</td>
<td>Performance</td>
</tr>
<tr>
<td>QoS-02</td>
<td>Reliability</td>
</tr>
<tr>
<td>QoS-03</td>
<td>Availability (front-end)</td>
</tr>
<tr>
<td>QoS-04</td>
<td>Availability (back-end)</td>
</tr>
<tr>
<td>Sof-Ver-01</td>
<td>Requirements Coverage</td>
</tr>
<tr>
<td>Sof-Ver-02</td>
<td>Open Defects</td>
</tr>
<tr>
<td>Indicator</td>
<td>Type target</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>Sof-Ver-03</td>
<td>All Defects</td>
</tr>
<tr>
<td>Sof-Int-01</td>
<td>Number of Enablers</td>
</tr>
<tr>
<td>Sof-Int-02</td>
<td>Number of successful uses by features</td>
</tr>
<tr>
<td>Sof-Reu-01</td>
<td>Documentation</td>
</tr>
<tr>
<td>Sof-Reu-02</td>
<td>Extensibility</td>
</tr>
<tr>
<td>Sof-Reu-03</td>
<td>Intellectual Property</td>
</tr>
<tr>
<td>Sof-Reu-04</td>
<td>Modularity</td>
</tr>
<tr>
<td>Sof-Reu-05</td>
<td>Packaging</td>
</tr>
<tr>
<td>Sof-Reu-06</td>
<td>Portability</td>
</tr>
<tr>
<td>Sof-Reu-07</td>
<td>Standards Compliance</td>
</tr>
<tr>
<td>Sof-Reu-08</td>
<td>Support</td>
</tr>
<tr>
<td>Indicator Id</td>
<td>Name</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>Sof-Reu-09</td>
<td>Verification and Testing</td>
</tr>
<tr>
<td>Ehe-eth-01</td>
<td>Privacy impact assessment (PIA)</td>
</tr>
<tr>
<td>Ehe-Imp-01</td>
<td>General Satisfaction</td>
</tr>
<tr>
<td>Ehe-Imp-02</td>
<td>Accessibility</td>
</tr>
<tr>
<td>Ehe-Imp-03</td>
<td>Adhereability</td>
</tr>
<tr>
<td>Ehe-Imp-04</td>
<td>Affordability</td>
</tr>
<tr>
<td>Ehe-Imp-05</td>
<td>Authenticity</td>
</tr>
<tr>
<td>Ehe-Imp-06</td>
<td>Availability</td>
</tr>
<tr>
<td>Ehe-Imp-07</td>
<td>Efficiency</td>
</tr>
<tr>
<td>Ehe-Imp-08</td>
<td>Non-Clinical Effectiveness</td>
</tr>
<tr>
<td>Ehe-Imp-09</td>
<td>Empowerment</td>
</tr>
<tr>
<td>Ehe-Imp-10</td>
<td>Safety</td>
</tr>
</tbody>
</table>
### Indicator

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Type target</th>
<th>Generic enablers</th>
<th>Specific enablers</th>
<th>Platforms</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ehe-Imp-11</td>
<td>Trustability</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ehe-Int-01</td>
<td>Integration tested</td>
<td>100%</td>
<td></td>
<td>100%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Targeted profiles depend on SE*
6 References


Speeds,” April 1999.


[33] European Commission, “Proposal for a Regulation of the European Parliament and of the Council on the protection of individuals with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation),” 2012.


Annex A  QoE Evaluation Questionnaire

### About the User

1. From the FI-STAR applications, please select one that you have experienced.

   - [ ] Diabetes Share System for Diabetes Care in Tromsø, Norway
   - [ ] TeleCare Solution for Rehabilitation and State Monitoring in Krakow, Poland
   - [ ] Chronic Disease Treatment Assistance for COPD Treatment in Bologna, Italy
   - [ ] Management Solution for Bipolar Patient Treatment in Bilbao, Spain
   - [ ] CRP Solution for Cardiac Rehabilitation in Bucharest, Rumania
   - [ ] Operating Theatre Monitor for Operation Consumables Tracking in Munich, Germany
   - [ ] Drug Supply Manager for Reverse Drug Supply Chain in Leeds, United Kingdom

2. Please tell us what the feature you experienced is called:

3. For how long have you used the feature?

   - [ ] Less than ½ hour
   - [ ] ½-1 hours
   - [ ] 1-5 hours
   - [ ] 6-20 hours
   - [ ] More than 20 hours

### Overall Questions

4. Overall, how satisfied are you with the feature?

   - [ ] Excellent (5)
   - [ ] Good (4)
   - [ ] Fair (3)
   - [ ] Poor (2)
   - [ ] Bad (1)

   Please tell us why you feel that way:

5. Overall, how do you judge the feature?

   - [ ] Exceptional
   - [ ] Better than comparable products and features
   - [ ] Good-enough
   - [ ] Insufficient

   Please tell us why you feel that way:
6. Will you return to use the feature again?

☐ Yes  ☐ No

Please tell us why:

---

### Specific Questions

The next question is about response time. With response time we mean the time from when you interacted with the application until it did what it was supposed to do.

7. How do you rate the response time of the feature?

☐ Excellent (5)  ☐ Good (4)  ☐ Fair (3)  ☐ Poor (2)  ☐ Bad (1)

Comment:

---

The next two questions are about reliability. With reliability we mean that the application is doing what it is supposed to do.

8. Have you seen any errors while using the feature?

☐ No error  ☐ 1 error  ☐ More than 1 error

Comment:

9. How do you rate the reliability of the feature?

☐ Excellent (5)  ☐ Good (4)  ☐ Fair (3)  ☐ Poor (2)  ☐ Bad (1)

Comment:
Annex B  Detailed explanation of the Reuse readiness scale

The definitions below provide the evaluation grid to position the evaluated software against the RRLs’ areas, as listed in [23].

B.1  Documentation

Documentation consists of installation and developer guides, development methodologies and documentation of the support available, API specifications, commented code and build instructions, technical support instructions and support forums, technical manuals, libraries and tutorials, and reuse and deployment case studies. This documentation may be in various stages of development and accessibility, and may not have a clear audience defined.

Documentation enables potential adopters to determine whether the software addresses the need and informs adopters how to utilize the software and reduce the risks and costs of reuse. Documentation includes descriptions of interfaces and capabilities, information about the execution environment, and instructions for the consumer on the purpose of the asset and on ways in can be reused. Documentation also describes plans for subsequent releases and future development.

Defined levels are:

- **Level 1 (Little or no internal or external documentation available)**: source code is available, with little or no useful internal or external documentation.
- **Level 2 (Partially to fully commented source code available)**: source code is available and fully commented, but no other documentation is provided. It may be challenging for a good programmer to determine how to reuse the software.
- **Level 3 (Basic external documentation for sophisticated users available)**: for example, a README file, a “man” page, or command line usage examples. This type of documentation would be sufficient for a sophisticated user to figure out how to use the software, but probably not a general user.
- **Level 4 (Reference manual available)**: reference manual provides complete documentation on use of the software, but may not be easily approached or accessed by general users. Some documentation relevant to customization is available.
- **Level 5 (User manual available)**: user manual allows a “normal” or general user to understand how to use and possibly customize aspects of the software.
- **Level 6 (Tutorials available)**: step-by-step walkthroughs of how the software is customized and used in various scenarios, demos, etc. This makes it very easy to understand/teach the software and use it in a new project.
- **Level 7 (Interface guide available)**: documentation describes how to customize and interface the software with other software, programmatic interfaces, APIs, etc., so that it can more easily be embedded in a larger system.
- **Level 8 (Extension guide and/or design/developers guide available)**: an extension guide provides information on how to customize and add to the software, add plug-ins and the like, use internal programming “languages”, etc. A design/developers guide provides a description of internals, design documentation, internal documentation, etc. that is sufficient for someone
“skilled in the art” to contribute to the development of the software or take over maintenance of the software.

- **Level 9 (Documentation on design, customization, testing, use, and reuse is available):** all stages of the software engineering lifecycle are fully documented. This includes design and review artifacts, testing artifacts, customization, and regression tests. The documentation provided is easy to read/access and is appropriate for different categories of users.

### B.2 Extensibility

The implementation takes into consideration future growth and ease of extending function. A measure of the ability to extend a system and the level of effort required to implement the extension. Extensions, or expandability, can apply to re-engineering or during runtime.

Extensibility is an important dimension to be able to incorporate an asset and add to or modify its functionality.

Defined levels are:

- **Level 1 (No ability to extend or modify program behaviour):** source code is not available; execution parameters cannot be changed, and/or it is not possible to extend the functionality of the software, even for application contexts similar to the original application domain.

- **Level 2 (Very difficult to extend the software system, even for application contexts similar to the original application domain):** the software was not designed with extensibility in mind. While some level of documentation and/or source code is available, it is extremely difficult to extend the software. For cases where source code is available, the logical flow of code may be hard to follow, with few (if any) comments, and little to no cohesion.

- **Level 3 (Extending the software is difficult, even for application contexts similar to the original application domain):** minimal consideration to extensibility is included in the design, through use of methods such as object-oriented design or other tools which provide logical cohesion. Where source code is available, the software has some structure, but may have a high number of independent logical paths, minimal comments and documentation, and/or a low degree of cohesion.

- **Level 4 (Some extensibility is possible through configuration changes and/or moderate software modification):** consideration to extensibility to some range of application contexts is included in the design though means such as (a) use of configuration files, (b) isolation of configuration parameters and constants in clearly identified sections of source code (distinct from logic and display code), (c) some documentation of the effects of changes to these parameters and the allowed values for these parameters, and/or (d) effective use of programming practices designed to enable reuse, such as object oriented design.

- **Level 5 (Consideration for future extensibility designed into the system for a moderate range of application contexts; extensibility approach defined and at least partially documented):** the procedures for extending the software are defined, whether by source code modification (e.g., object-oriented design) or through the provision of some type of extension functionality (e.g., callback hooks or scripting capabilities). Where source code modification is part of the extension
plan, the software is well-structured, has a moderate to high level of cohesion, and has configuration elements clearly separated from logic and display elements. Internal and external documentation are sufficient to allow an experienced programmer to understand program flow and logic with moderate effort.

- **Level 6** (Designed to allow extensibility across a moderate to broad range of application contexts, provides many points of extensibility, and a thorough and detailed extensibility plan exists): the extensibility capability for the software is well defined, sufficient to enable an experienced developer generally familiar with the project to extend the software. That documentation should include clear information about the range of application contexts to which the software can be extended as well as potential limitations on expansion.

- **Level 7** (Demonstrated to be extensible by an external development team in a similar context): the software has been extended and applied to a similar application context to the original. This extension may have been done by an external team using extension documentation, by may have involved substantial assistance from the original development team members.

- **Level 8** (Demonstrated extensibility on an external program, clear approach for modifying and extending features across a broad range of application domains): the software has been extended by at least one group of users outside the original development group using existing documentation and with no assistance from the original development team.

- **Level 9** (Demonstrated extensibility in multiple scenarios, provides specific documentation and features to build extensions which are used across a range of domains by multiple user groups): the software is regularly extended externally by users across multiple applications using available documentation. There may be a library available of user-generated content for extensions.

### B.3 Intellectual Property Issues

A formal and documented explanation of the involved parties and roles, with binding statements describing any licensing mechanisms, ownership rights, restrictions, and user/consumer responsibilities related to the distribution and reuse of assets. The legal rights are established in accordance with the policies and laws of the organization that originally produced the software.

Potential adopters need to understand the intellectual property issues to know whether they have the authority to reuse the software.

Defined levels are:

- **Level 1** (Developers have been identified, but no rights have been determined): product developers have been identified and their responsibilities have been determined, but they have not considered or determined the rights for the product.

- **Level 2** (Developers are discussing rights that comply with their organizational policies): relevant policies of developers have been reviewed for applicability to intellectual property rights, but no agreements have been proposed. Rights are not specified.
• **Level 3 (Rights agreements have been proposed to developers):** each developer has received a draft intellectual property rights agreement that would result from cooperative activities with other developers. Rights are not specified.

• **Level 4 (Developers have negotiated on rights agreements):** developers have reviewed proposals from each of the other developers and have proposed an agreement that addresses any potential conflicts in the proposed intellectual property rights and responsibilities for development. A limited rights statement has been drafted and developers may be contacted to negotiate rights for reuse.

• **Level 5 (Agreement on ownership, limited reuse rights, and recommended citation):** developers have agreed on proposed ownership, limited intellectual property rights for reuse, and responsibilities. Order of developers’ names, recommended citation, and agreements have been formalized. Developers may be contacted to obtain formal statements on restricted rights for reuse.

• **Level 6 (Developer list, recommended citation, and rights statements have been drafted):** agreements on development responsibilities, the list of developers, a recommended citation, and intellectual property rights statements, offering limited rights for reuse have been drafted and are included in package. Developers may be contacted to obtain formal statements on restricted rights or to negotiate additional rights.

• **Level 7 (Developer list and limited rights statement included in product prototype):** a list of developers, recommended citation, and intellectual property rights statements, including copyright or ownership statements, are embedded in the source code of the product, in the documentation, and in the expression of the software upon execution. These include any legal language that has been approved by all parties or their representatives, machine-readable code expressing intellectual property, and concise statements in language that can be understood by laypersons, such as a pre-written, recognizable license. Brief statements are available describing limited rights, restrictions, and conditions for reuse. Developers may be contacted to negotiate additional rights.

• **Level 8 (Recommended citation and intellectual property rights statement included in product):** all parties have reviewed the list of developers, recommended citation, and intellectual property rights statements, including limited rights for reuse, in the product to ensure that all interests are represented and that the statements conform to their institutional policies and agreements. Brief statements are available describing unrestricted rights and any conditions for reuse. Developers may be contacted to obtain formal rights statements.

• **Level 9 (Statements describing unrestricted rights, recommended citation, and developers embedded into product):** multiple statements are embedded into the product describing unrestricted rights and any conditions for reuse, including commercial reuse, and the recommended citation. The list of developers is embedded in the source code of the product, in the documentation, and in the expression of the software upon execution. The intellectual property rights statements are expressed in legal language, machine-readable code, and in concise statements in language that can be understood by laypersons, such as a pre-written, recognizable license.
B.4 Modularity

Modularity is a software design technique that increases the extent to which software is composed from separate components, called modules. Conceptually, modules represent a separation of and encapsulation of concern, purpose, and function, and they improve maintainability and reusability.

Modular assets generally are easier to synthesize and extend.

Defined levels are:

- **Level 1 (Not designed with modularity)**: research or prototype-grade code written with no designs for organizing code in terms of functionality for modularity or reuse.
- **Level 3 (Modularity at major system or subsystem level only)**: no clear distinctions between generic and solution-specific functionality; few internal functions accessible by external programs (i.e., closed architecture), limited distinction between visible functions; code is organized into a primary system that provides general functionality and one or two subsystems that each provide multiple, unrelated, functions; code within each module contains many independent logical paths.
- **Level 5 (Partial segregation of generic and specific functionality)**: top to bottom structuring into individual components that provide functions or services to outside entities (i.e., open architecture); internal functions or services documented, but not consistently; modules have been created for generic functions, but modules have not been created for all of the specified functions; code within each module contains many independent logical paths.
- **Level 7 (Clear delineations of specific and reusable components)**: organization of all components into libraries or service registries; consistent documentation of all libraries as APIs or standard web service interfaces; modules have been created for all specified functions and organized into libraries with consistent features within interfaces; code within each module contains many independent logical paths.
- **Level 9 (All functions and data encapsulated into objects or accessible through web service interfaces)**: all functions and data encapsulated into objects or accessible through web service interfaces; consistent error handling; use of generic extensions to program languages for stronger type checking and compilation-time error checking; services available externally, e.g., in “third-party” service workflows; code within each module contains few independent logical paths.

B.5 Packaging

Packaging pertains to the technologies, standards, and procedures related to gathering, organizing, assembling, and compressing the parts of a software system and distributing it as a collection. Packaging is important to ensure completeness, to allow distribution, and to simplify the installation of the asset.

Defined levels are:
B.6 Portability

The independence of an asset from platform-specific technologies.

Portability refers to two components: software consisting of source code that can be compiled for various computing platforms; software executables that can be executed on various platforms.

The ability to be installed or executed on various platforms maximizes reuse potential and increases the flexibility and (re-)usability of the asset and its applications.

Defined levels are:

- **Level 1 (The software is not portable)**: no source code or instructions for customization are provided. Executable binaries are provided and there are known severe limitations for running it on the hardware or operating system. There is only minimal information on installation or use. There is no information on porting to another platform or application.

- **Level 2 (Some parts of the software may be portable)**: some source code is provided with some internal and external documentation. Binaries are provided and there is some documentation on how to install the software. There is no useful information on porting. Porting is prohibitively expensive, but some portions (e.g. modules, functions) of the code may be portable.

- **Level 3 (The software is only portable with significant costs)**: the complete source code is available, without external dependencies that are portable, but the software cannot be ported without significant changes to the software or the target context. Documentation on porting the code to another platform or application is missing or deficient. Porting would not be practical or cost effective.

- **Level 4 (The software may be portable at a reasonable cost)**: the cost benefits of using the software slightly outweigh the cost of developing new software. Documentation is barely sufficient, but may contain some useful information on porting to another platform or application. Porting will nonetheless require significant effort. Only at this level is it generally worth considering porting the software.
• **Level 5 (The software is moderately portable):** the software can be ported with only relatively small changes necessary to the context or the software itself. Documentation on porting exists and is complete, but requires considerable effort and expertise. Some rudimentary understanding of the underlying software or the target system may be necessary.

• **Level 6 (The software is portable):** the software can be ported to most major systems without modification. The documentation, however, addresses porting to a large number of systems that are identified. Any modifications needed to port the software to these systems are well described in the documentation and would be relatively easy to implement.

• **Level 7 (The software is highly portable):** the software can be ported to all but the most obscure or obsolete systems without modification. The documentation is complete and thorough. No changes to the software are necessary and the effort to port the software is minimal.

• **Level 9 (The software is completely portable):** the software can be ported to all systems since it runs on an application layer rather than on the underlying operating system layer. Such software is written in languages Java, C#, etc. In theory at least, the software will run on any system in which the appropriate application layer has been installed.

### B.7 Standards Compliance

Concerning commonly accepted criteria, models, patterns and/or specifications have been followed in the creation of a reusable asset; and at what level the asset complies with the standard. By complying with accepted standards, the asset has increased potential for adoption.

Defined levels are:

• **Level 1 (No standards compliance):** neither the software nor the software development process adheres to any identified standards other than those inherent in the software languages employed.

• **Level 2 (No standards compliance beyond best practices):** the software and software development process adhere, at least in part, to some common best practices, but do not identify or claim compliance with any recognized standard.

• **Level 3 (Some compliance with local standards and best practices):** the software and software development process comply with standards and best practices defined locally by the development organization.

• **Level 4 (Standards compliance, but incomplete and untested):** the software and software development process attempt to comply with recognized standards, but without verification. Standards compliance is thus untested and may not be complete.

• **Level 5 (Standards compliance with some testing):** the software and software development process comply with recognized standards, but verification of compliance is incomplete. Standards compliance may not be followed by all components.

• **Level 6 (Verified standards compliance with proprietary standards):** the software and software development process comply with specific and proprietary
standards (such as Windows GUI) and compliance with those standards has been verified through testing.

- **Level 7 (Verified standards compliance with open standards)**: the software and software development process comply with specific open standards and compliance with those standards has been verified through testing.

- **Level 8 (Verified standards compliance with recognized standards)**: the software and software development process comply with internationally recognized standards such as W3C, XML, XHTML, WAI, IP for Web; or ANSI/ISO (C/C++), JCP (Java), for software; and CMMI, IEEE Software Engineering Standards for development process. Standards compliance has been verified through testing, but not by an independent testing organization.

- **Level 9 (Independently verified standards compliance with recognized standards)**: the software and software development process comply with internationally recognized standards. Independent and documented standards compliance verification is included with the software. The development organization maintains standards compliance in its development process through regular testing and certification from an independent group.

### B.8 Support

Technical support exists, in the form of various communication methods with the asset’s developers, documentation/knowledge bases, user communities, support level agreements, and online forums. A release strategy and plan for patches and versions has been created. Support provisions expertise to assist in maintenance, evolution, extension and issue resolution.

**Defined levels are:**

- **Level 1 (No support available)**: the original developer of the code is not known, not locatable, or is refusing support.

- **Level 2 (Minimal support available)**: there is known contact information available for the original developer(s) and they are willing to provide minimal, occasional support.

- **Level 3 (Some support available)**: contact information is available and there is a willingness to provide some support infrequently, without guarantees. This may include things such as providing makefiles or different flavors of the code for different contexts.

- **Level 4 (Moderate systematic support is available)**: latest updates/patches are usually made available. Support is available, but may be intermittent.

- **Level 5 (Support provided by an informal user community)**: there is an informal user community that provides answers, for example, via a Web site FAQ.

- **Level 6 (Formal support available)**: support is centralized in a web site containing relevant resources, answers to FAQ, and other useful information.

- **Level 7 (Organized/defined support by developer available)**: there is organized and defined support by the developer with email/telephone help desk and links to case studies and other relevant information. No continuity of support implied.

- **Level 8 (Support available by the organization that developed the asset)**: the support is by an organization and is well defined with frequent updates, releases,
etc., and help desk. Continuity of support is implied. Support may be free or fee-based and may be offered by a third party.

- **Level 9 (Large user community with well-defined support available):** this may include resources such as a help desk, a Web site for the latest information, an active discussion group willing to answer questions, frequent patches and updates as well as consolidation of changes by the community. One example would be the Linux operating system.

### B.9 Verification and Testing

This can be realized through the provision of test material, requirements compliance, proper function, and usability (robustness). Tests documented, results analyzed and published, and fixes and enhancements applied. Sufficient verification and testing increases the accuracy and confidence and reduces potential risks and costs of reuse.

Defined levels are:

- **Level 1 (No testing performed):** ideas have been translated into software development. Examples might include studies of development languages, prototype, or diagram of interface. Requirements have not been verified, and there is no formal test mechanism in place.

- **Level 2 (Software application formulated and unit testing performed):** software application compiles, and executes with known inputs. For example, a prototype application where there is no testing or validation to support the software, but only testing to demonstrate a prototype. Requirements may not be finalized yet, or overall testability of the software determined.

- **Level 3 (Testing includes testing for error conditions and handling of unknown input):** software applications have been “white box” tested. This includes both known and unexpected inputs to the application. This level of testing has been incorporated into the build and/or deployment mechanism.

- **Level 4 (Software application demonstrated in a laboratory context):** following successful testing of inputs and outputs, the testing has integrated an application to establish that the “pieces” will work together to achieve concept-enabling levels. This validation has been devised to support the concept that was formulated earlier, and is consistent with the requirements of potential system applications. The validation is relatively “low-fidelity” compared to the eventual system – it could be composed of ad hoc discrete components in a laboratory; for example, an application tested with simulated inputs.

- **Level 5 (Software application tested and validated in a laboratory context):** the fidelity of the software application testing has not been demonstrated. The software application must be integrated with reasonably realistic supporting elements so that the total application (component level, sub-system level, or system level) can be tested in a “simulated” or somewhat relevant context. At this level, issues such as scalability, load testing, and security are addressed when applicable.

- **Level 6 (Software application demonstrated in a relevant context):** the fidelity of the software application testing has not been demonstrated. The software application must be integrated with existing elements and interfaces so that the total application (component level, sub-system level, or system level) can be
tested and validated in a relevant context. At this level, issues such as number of users and operational scenarios, as well as load testing and security are addressed if applicable.

- **Level 7 (Software application tested and validated in a relevant context):** the software application testing meets the requirements of the application that apply to the software when it is to be delivered or installed. The software application has been tested in the lab so that the application can be validated as if the software were delivered for use in another context. At this level, all issues have been resolved regarding security and operational scenarios.

- **Level 8 (Software application “qualified” through test and demonstration (meets requirements) and successfully delivered):** the software has passed testing and meets all requirements of the software, with the additional testing of the software delivery and installation for various applications.

- **Level 9 (Actual software application tested and validated through successful use of application output):** demonstrable that for any application of the software, testing shows the software meets all defined requirements.