## D2.1 Process baseline definition

**State-of-the-practice (V1.0)**

<table>
<thead>
<tr>
<th>Programme</th>
<th>H2020</th>
</tr>
</thead>
<tbody>
<tr>
<td>Funding scheme</td>
<td>RIA - Research and Innovation action</td>
</tr>
<tr>
<td>Topics</td>
<td>ICT-10-2016 - Software Technologies</td>
</tr>
<tr>
<td>Project number</td>
<td>732253</td>
</tr>
<tr>
<td>Project name</td>
<td>Quality-aware rapid software development</td>
</tr>
<tr>
<td>Project duration</td>
<td>1\textsuperscript{st} November 2016 – 31\textsuperscript{st} October 2019</td>
</tr>
<tr>
<td>Project website</td>
<td><a href="http://www.q-rapids.eu">www.q-rapids.eu</a></td>
</tr>
<tr>
<td>Project WP</td>
<td>WP2 – Quality-aware Rapid Software Development Process</td>
</tr>
<tr>
<td>Project Task</td>
<td>Task 2.1 – Process baseline definition</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Deliverable type</th>
<th>X</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEM</td>
<td>Demonstrator, pilot, prototype</td>
<td></td>
</tr>
<tr>
<td>DEC</td>
<td>Websites, patent fillings, videos, etc.</td>
<td></td>
</tr>
<tr>
<td>OTHER</td>
<td>Material that does not belong to any specified category</td>
<td></td>
</tr>
<tr>
<td>ETHICS</td>
<td>Ethics requirement</td>
<td></td>
</tr>
</tbody>
</table>

| Contractual delivery | 28/4/2017 |
| Delivered | 28/4/2017 |
| Responsible beneficiary | Organisation |

<table>
<thead>
<tr>
<th>Dissemination level</th>
<th>X</th>
<th>PU</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO</td>
<td>Confidential, only for members of the consortium (including the Commission Services)</td>
<td></td>
</tr>
<tr>
<td>EU-RES</td>
<td>Classified Information: RESTREINT UE (Commission Decision 2005/444/EC)</td>
<td></td>
</tr>
<tr>
<td>EU-CON</td>
<td>Classified Information: CONFIDENTIEL UE (Commission Decision 2005/444/EC)</td>
<td></td>
</tr>
</tbody>
</table>
Major Version Plan

<table>
<thead>
<tr>
<th>Version no.</th>
<th>Month</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1.0</td>
<td>M6</td>
<td>28/4/2017</td>
</tr>
<tr>
<td>V2.0</td>
<td>M15</td>
<td>31/01/2017</td>
</tr>
</tbody>
</table>
### Version history

<table>
<thead>
<tr>
<th>Version no.</th>
<th>Date</th>
<th>Description</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0.1</td>
<td>24/03/2017</td>
<td>Delivery template deployed</td>
<td>Pertti Karhapää, UOULU</td>
</tr>
<tr>
<td>V0.2</td>
<td>27/03/2017</td>
<td>Constructed the structure of the deliverable, updated sections with information about what they will include together with example text, included example tables and example figures</td>
<td>Pertti Karhapää, UOULU</td>
</tr>
<tr>
<td>V0.3</td>
<td>31/03/2017</td>
<td>Changes after UoO internal review</td>
<td>Pertti Karhapää, UOULU</td>
</tr>
<tr>
<td>V0.4</td>
<td>31/03/2017</td>
<td>Minor changes</td>
<td>Pertti Karhapää, UOULU</td>
</tr>
<tr>
<td>V0.5</td>
<td>12/04/2017</td>
<td>Adding content</td>
<td>Pertti Karhapää, Woubshet Behutiye, UOULU</td>
</tr>
<tr>
<td>V0.6</td>
<td>13/04/2017</td>
<td>Improving use case relevant information for review by the use case companies</td>
<td>Pertti Karhapää, Woubshet Behutiye, UOULU</td>
</tr>
<tr>
<td>V0.7</td>
<td>18/04/2017</td>
<td>Applying deliverable template V6</td>
<td>Pertti Karhapää, UOULU</td>
</tr>
<tr>
<td>V0.8</td>
<td>19/04/2017</td>
<td>Adding aggregated development process, QRs management practices, and challenges and areas of improvement. Moving UC provider companies’ context information and process descriptions to non-public annexes. Adding Interview script to Annex A.</td>
<td>Pertti Karhapää, Woubshet Behutiye, UOULU</td>
</tr>
<tr>
<td>V0.85</td>
<td>19/04/2017</td>
<td>General revision</td>
<td>Pilar Rodríguez, UOULU</td>
</tr>
<tr>
<td>V0.9</td>
<td>19/04/2017</td>
<td>UOULU internal comments addressed</td>
<td>Pertti Karhapää, UOULU</td>
</tr>
<tr>
<td>V0.10</td>
<td>20/04/2017</td>
<td>Corrected wrongly added RQs. Adding missing content to 4.1 and 4.3</td>
<td>Pertti Karhapää, UOULU</td>
</tr>
<tr>
<td>V0.11</td>
<td>21/04/2017</td>
<td>Formatting tables, correcting references, adding missing content to sections 1, 2.2, 4.2, 4.3, and Annex B.</td>
<td>Pertti Karhapää, Woubshet Behutiye, UOULU</td>
</tr>
<tr>
<td>V0.12</td>
<td>21/04/2017</td>
<td>Changed sections 4.2 and 4.3</td>
<td>Woubshet Behutiye, UOULU</td>
</tr>
<tr>
<td>V0.13</td>
<td>21/04/2017</td>
<td>General revision</td>
<td>Pilar Rodríguez, UOULU</td>
</tr>
<tr>
<td>V0.14</td>
<td>21/04/2017</td>
<td>Accepted changes and addressed comments</td>
<td>Pertti Karhapää, UOULU</td>
</tr>
<tr>
<td>V0.15</td>
<td>21/04/2017</td>
<td>Accepted changes and addressed comments</td>
<td>Woubshet Behutiye, UOULU</td>
</tr>
<tr>
<td>V0.16</td>
<td>26/04/2017</td>
<td>Included major version table, addressing comments by reviewers</td>
<td>Pertti Karhapää, UOULU</td>
</tr>
<tr>
<td>V0.17</td>
<td>26/04/2017</td>
<td>Changed and updated state-of-the-art</td>
<td>Woubshet Behutiye, UOULU</td>
</tr>
<tr>
<td>V0.18</td>
<td>26/04/2017</td>
<td>General revision</td>
<td>Pilar Rodríguez, UOULU</td>
</tr>
<tr>
<td>V0.19</td>
<td>27/04/2017</td>
<td>Included annex list. Updated section 1.5 and table 1. Addressed comments in state-of-the-art, challenges, and conclusion. Updated annexes A, F, and G</td>
<td>Woubshet Behutiye, UOULU</td>
</tr>
<tr>
<td>V0.20</td>
<td>27/04/2017</td>
<td>Updated major version plan. Updated table 1. Addressed comments in section 4. Changed section 3 so table fit on one page and figure on own page. Complete review of all sections.</td>
<td>Pertti Karhapää, UOULU</td>
</tr>
<tr>
<td>Version</td>
<td>Date</td>
<td>Changes</td>
<td>Authors</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
<td>--------------------------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>V0.21</td>
<td>28/04/2017</td>
<td>Minor corrections and improvements. Improved Figures 2 and 6</td>
<td>Woubshet Behutiye, Pertti Karhapää, OUULU</td>
</tr>
<tr>
<td>V0.22</td>
<td>28/04/2017</td>
<td>Final corrections</td>
<td>Pilar Rodriguez, Pertti Karhapää, Woubshet Behutiye, OUULU</td>
</tr>
<tr>
<td>V1.0</td>
<td>28/04/2017</td>
<td>Uploaded for Consortium reviewers</td>
<td>Pilar Rodriguez, OUULU</td>
</tr>
</tbody>
</table>
Authors

<table>
<thead>
<tr>
<th>Organisation</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>UOULU</td>
<td>Pertti Karhapää</td>
</tr>
<tr>
<td>UOULU</td>
<td>Woubshet Behutiye</td>
</tr>
<tr>
<td>UOULU</td>
<td>Pilar Rodríguez</td>
</tr>
</tbody>
</table>

Reviewers

<table>
<thead>
<tr>
<th>Organisation</th>
<th>First name last name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fraunhofer</td>
<td>Liliana Guzmán</td>
</tr>
<tr>
<td>Softeam</td>
<td>Andrey Sadovykh</td>
</tr>
</tbody>
</table>

Disclaimer

The work associated with this report has been carried out in accordance with the highest technical standards and the Q-Rapids partners have endeavoured to achieve the degree of accuracy and reliability appropriate to the work in question. However, since the partners have not control over the use to which the information contained within the report is to be put by any other party, any other such party shall be deemed to have satisfied itself as to the suitability and reliability of the information in relation to any use, purpose or application.

Under no circumstances will any of the partners, their servants, employees or agents accept any liability whatsoever arising out of any error or inaccuracy contained in this report (or any further consolidation, summary, publication or dissemination of the information contained within this report) and/or the connected work and disclaim all liability for any loss, damage, expenses, claims or infringement of third party rights.
**Definition of the key terms and abbreviations**

<table>
<thead>
<tr>
<th>A term or an abbreviation (alphabetical order)</th>
<th>Explanation of the term or the abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASD</td>
<td>Agile software development</td>
</tr>
<tr>
<td>CD</td>
<td>Continuous delivery</td>
</tr>
<tr>
<td>CI</td>
<td>Continuous integration</td>
</tr>
<tr>
<td>Dm.n</td>
<td>Deliverable</td>
</tr>
<tr>
<td>DoD</td>
<td>Definition of Done</td>
</tr>
<tr>
<td>FR</td>
<td>Functional requirement</td>
</tr>
<tr>
<td>NFR</td>
<td>Non-functional requirement</td>
</tr>
<tr>
<td>QR</td>
<td>Quality requirement</td>
</tr>
<tr>
<td>RE</td>
<td>Requirements engineering</td>
</tr>
<tr>
<td>RQ</td>
<td>Research question</td>
</tr>
<tr>
<td>RSD</td>
<td>Rapid software development</td>
</tr>
<tr>
<td>SLR</td>
<td>Systematic literature review</td>
</tr>
<tr>
<td>SW</td>
<td>Software</td>
</tr>
<tr>
<td>Tx.x</td>
<td>Task</td>
</tr>
<tr>
<td>UC</td>
<td>Use case</td>
</tr>
<tr>
<td>WP</td>
<td>Work package</td>
</tr>
</tbody>
</table>
Contents
The list of tables ................................................................................................................. 8
The list of figures ............................................................................................................... 9
The list of Annexes ........................................................................................................... 10
Executive Summary ......................................................................................................... 11
1. Introduction .................................................................................................................. 12
   1.1 Motivation ............................................................................................................... 12
   1.2 Intended audience ................................................................................................. 12
   1.3 Scope ..................................................................................................................... 12
   1.4 Relation to other deliverables ............................................................................. 12
   1.5 Structure of the deliverable ................................................................................ 12
2. State-of-the-art ............................................................................................................. 14
3. Use cases’ analysis ....................................................................................................... 16
   4.1 Context information .............................................................................................. 19
   4.2 General software development process .............................................................. 19
   4.3 Managing quality requirements ....................................................................... 21
   4.4 Challenges and areas of improvement ............................................................... 23
5. Implications for the Q-Rapids framework .................................................................. 25
Conclusions ...................................................................................................................... 27
References ....................................................................................................................... 28
Annex A – SLR protocol ................................................................................................. 30
Annex B – Interview script ............................................................................................ 32
Annex C – UC1 ............................................................................................................... 35
Annex D – UC2 ............................................................................................................... 36
Annex E – UC3 ............................................................................................................... 37
Annex F – UC4 ............................................................................................................... 38
Annex G – Process related terms ................................................................................ 39
The list of tables

Table 1: Interviewees’ profiles .............................................................................................................. 17
Table 2. SLR - Data properties ............................................................................................................ 31
The list of figures

Figure 1. Screenshot of coding scheme in NVivo 11 for coding transcribed interviews. .......................... 18
Figure 2. Aggregated view of the development processes in the UCs......................................................... 21
Figure 3. Vision of Q-Rapids framework........................................................................................................ 25
The list of Annexes

NOTE! Because of confidentiality issues annexes C, D, E, and F are privately available only for Q-Rapids partners and Commission reviewers. Once this document is made publicly available, these annexes must be deleted from the document.

Annex A – SLR protocol
Annex B – Interview script
Annex C – UC1
Annex D – UC2
Annex E – UC3
Annex F – UC4
Annex G – Process related terms
Executive Summary
This deliverable provides a descriptive analysis of the current processes to manage quality requirements (QRs) applied in the use cases (UCs) provided by the Q-Rapids’ industry partners. It contains: 1) a description of the processes applied in each UC, 2) an aggregated view of all UCs’ processes, which conforms the project’s process baseline, 3) description of the problems we found in the UCs for managing QRs, and 4) improvement areas as potential opportunities for Q-Rapids, from work package (WP) 2’s point of view. The document provides also a descriptive analysis of the state-of-the-art on managing QRs in agile software development (ASD) and rapid software development (RSD).
1. Introduction
This section presents the purpose of this deliverable, intended audience, the scope of the deliverable, relation to other deliverables, and, finally, the structure of the deliverable.

1.1 Motivation
The purpose of this deliverable is to provide a descriptive analysis of state-of-the-art of QRs management in agile software development (ASD) and rapid software development (RSD), and state-of-the-practice of QRs management in the UCs provided by Q-Rapids’ companies. Further, the purpose is to describe the challenges and areas of improvement in current practices for managing QRs. The outcome of the document is a baseline process for QRs management together with implications for the Q-Rapids framework that will be implemented in the project.

1.2 Intended audience
As this deliverable is a public document, its intended audience are both the public audience as well as the Q-Rapids members, including both industry and research partners. For Q-Rapids members, this document sets the basis on which to build the Q-Rapids software development process. Any Q-Rapids member can use this document to get informed about current processes for managing QRs in the project’s UCs. As the Q-Rapids development process will be incrementally developed, they can contribute in the next development iterations. Because of confidentiality issues, annexes giving UC information are only privately available for Q-Rapids partners and Commission reviewers. Once this document is made publicly available, these annexes must be deleted from the document.

1.3 Scope
The scope of this deliverable is the initial phases of the Q-Rapids project, setting up the baseline process for the following tasks. Concretely, this deliverable defines the baseline for the lifecycle process that will be defined for managing QRs, which will be the outcome of work package (WP) 2. The baseline process will also serve as basis for the evaluation of the Q-Rapids framework. Furthermore, problematic areas related to QRs management in the project’s UCs are identified and defined in this document. These problematic areas will be later considered for designing the Q-Rapids quality-aware software development process (tasks T2.3 and T2.4), and will serve as input to WP1 and WP3. In addition, areas of improvement will serve as input for the definition of the evaluation scope. However, as an iterative activity, it is expected that this document will evolved during the course of the project (concretely, it will be updated at M15. Further updates will be considered if needed).

1.4 Relation to other deliverables
This deliverable works as a basis for coming tasks and deliverables in Q-Rapids WP2. For deliverable D2.2 Baseline process metrics to support the identification of metrics to analyze the Q-Rapids process performance, for D2.3 and D2.4 Construction and validation of the Q-Rapids process as the basis to build on, for D2.5 Q-Rapids process description for comparing performance and quality to baseline process, and for the D2.6 Q-Rapids process final report.

1.5 Structure of the deliverable
The rest of the document is structured as follows: Section 2 presents the state of the art in the scientific literature on managing QRs in ASD and RSD. Section 3 describes analysis of the UCs, the research method, and steps followed in the analysis. Section 4 presents the state of the practice in the UCs. It presents the aggregated view of UCs processes for developing software and, more concretely, managing QRs. An analysis
of the challenges and areas of improvement identified by the UCs is also presented in Section 4. Section 5 depicts implications of the findings on how to build the Q-Rapids framework upon the process baseline description presented in Section 4. Finally, conclusions of the deliverable are presented at the end of the document.
2. State-of-the-art

We analyzed the scientific literature on QRs management in ASD and RSD in order to understand the state-of-the-art on the area. We found that there is an increasing number of studies reporting challenges and solutions for managing QRs in ASD and RSD (e.g. Martakis et al. [1], Savolainen et al. [2], Chen [3], Cannizo et al. [4], Farid et al. [5]). Likewise, the findings from recent secondary studies on agile requirements engineering [6], [7], [8], [9], [10], [11] highlight the significance of the topic, and emphasize that managing QRs in the context of ASD is highly challenging. For instance, Inayat et al. [8] found that little attention is given to QRs (also referred to as non-functional requirements, NFRs) while investigating challenges and practices of requirements engineering (RE) in ASD. According to the authors, QRs are neglected in ASD since user stories describe only system and product features. QRs were also associated to delay and rework. The authors found some practices for dealing with QRs in ASD. The practices were mainly related to testing activities, including system quality test (performance and load testing), scenario and exploratory testing practices for QRs, independent inspection testing during the lifecycle of a system, and testing of functional requirements (FRs) (e.g. feature and system acceptance tests). However, the authors concluded that further empirical investigation to evaluate the usefulness of existing proposals and, in general, research on QRs in ASD are needed.

Overall, the software engineering literature reflects numerous challenges when managing QRs in the context of ASD and RSD. For instance, Heikkilä et al. [7] identified that the negligence of QRs and the reliance on tacit knowledge for implementing QRs are challenges in ASD. Furthermore, user stories seem to be insufficient to describe QRs. Difficulties in scaling user stories for documenting QRs in large and complex systems and lack of support for traceability of QRs were other challenges identified in this study.

Another secondary study by Schön et al. [10] presents also a brief analysis of QRs (referred as NFRs in the paper) in ASD. However, the particular focus of the study is on investigating stakeholder and user involvement in RE of ASD. Similar to the findings of other systematic literature reviews (SLRs) [7], [8], [11], the negligence of QRs in ASD is identified as a challenge in this study. The lack of earlier consideration of QRs produces additional rework and bottlenecks. In addition, estimation and measurement of QRs (e.g. user experience metrics and security policies) were identified as open challenges as well. With regards to strategies for managing QRs in ASD, this secondary study found: 1) the Agile Framework For Integrating Non-functional requirements Engineering (AFFINE framework), which introduces a role for NFR into Scrum [12], and 2) the Non-functional Requirements Modelling for Agile Process (NORMAP), where a taxonomy is used to classify requirements as functional or non-functional [13].

A recent SLR by Alsaqaf et al. [11] investigated agile practices used to engineer QRs in the context of large scale distributed agile projects, and challenges of QRs and existing solutions for managing QRs in ASD in general. The authors found that there are some specific practices for managing QRs such as security, safety, compliance and usability in the literature. For instance, the Security-Enhanced Agile process by Baca et al. [14] that introduces security metrics and risk analysis in the agile RE process and additional roles such as security manager, security master and security architect for integrating security features (that are QRs) in ASD. Similar to the findings of [7], [8], the authors identified that QRs are often neglected in ASD. Other challenges for managing QRs were identified as follows: focus on delivery functionality at the cost of architecture, ignoring predictable architecture requirements, inability of user stories to document QRs and requirements dependencies, product owner’s lack of knowledge on product quality and aspects related to QRs, and dependence on the product owner as a single source of requirements gathering.

Martakis et al. [1] identified also challenges related to QRs in ASD, while investigating cross-cutting requirement dependencies in agile projects. Cross-cutting requirements include QRs such as security, and internationalization (multiple language support). The authors identified that their interviewees had difficulty
in capturing these QRs by using user stories. In addition, they had difficulties in showing the business value of QRs to their customers. Similarly, Savolainen et al. [2] identify difficulties in managing architecturally significant requirements (e.g. performance and user interface) in ASD. Farid and Mitropoulos [5] propose the NORPLAN (Non-functional Requirements Planning for agile process) to improve the planning and prioritization of NFRs.

In the context of RSD, Cannizo et al. [4] describes that the implementation of continuous integration (CI) can address QRs (particularly robustness and performance). According to the authors, robustness and performance builds through CI enabled identifying issues that were not visible in acceptance tests. Similarly, Chen [3] shows how CI enhanced releases’ reliability. Continuous delivery (CD) enabled to identify errors in deployment earlier through frequent testing.

Overall, we found that the scientific literature acknowledges the importance of properly managing QRs in ASD and RSD. The premise is that faster and more frequent release cycles should not compromise quality. However, ASD still proves to offer little support to the integration and management of quality. Empirical evidence shows challenges when managing quality in ASD and RSD process. For instance, the high orientation towards customers makes functionality and fast delivery cycles to play a relevant role that may provoke an overlook of QRs and quality management activities in general. Further, as the product evolves, software quality may be neglected due to cost and time constraints. Hence, maintenance costs may rise and the development of functionalities may take longer. Although the literature includes some initial solutions for managing QRs in ASD (e.g. AFFINE framework, NORMAP, NORPLAN, CI), the reported proposals often lack empirical validation [8] [10] [11], making unclear their usefulness. In this regard, WP2 aims to introduce a quality aware RSD process, which will allow a proper management of QRs in the context of ASD and RSD and will be empirically validated during the project. Solutions found in the literature will be considered as potential solutions that could be incorporated into the Q-Rapids framework.

In addition to this descriptive analysis of the state-of-the-art on managing QRs in ASD and RSD, we are conducting a SLR with the scientific aim of writing a journal publication. The protocol of the SLR is available in Annex A.
3. Use cases’ analysis

We analysed the state-of-the-practice on managing QRs based on the project’s UCs in the form of separate case studies (one case study per UC). Runeson and Höst [15] guidelines has been adapted for designing and conducting the case studies. While designing the case studies, the objective was determined, research questions (RQs) were formulated and data collection and analysis strategies were planned beforehand. Our main objective was to understand the state of the practice of QRs management in the four UCs we identified in the Q-Rapids consortium. We formulated RQs to identify current software development processes, QRs management practices, challenges of QRs management, and areas of improvement in the UCs.

To answer our RQs, we collected data using semi-structured interviews with relevant stakeholders in each UC. A pre-questionnaire was sent to the champions (contact persons) of the UCs, prior to the interviews, to more accurately set-up the scope of the interviews and prepare the interview script (see Appendix B). In the selection strategy, we considered roles that directly and indirectly impact the QRs management process in the UCs. Our champion in each company helped us identify the persons to be interviewed. In the data collection phase, triangulation has been applied by including information gathered not only from the interviews, but also from the quality workshop conducted during the company visits, and documents provided by the UC providers. The pre-questionnaire was also used to cross-check some parts of the analysis. Interviewees’ profiles are presented in Table 1.

Overall, we conducted 12 interviews, which were recorded. Three from UCs 1 and 2, four from UC3 and two from UC4. All the recordings were sent to a professional transcription service to be transcribed, and while waiting for the transcriptions to arrive we did a preliminary analysis by listening to the interview recordings to get a general overview of the development processes in the UCs. When the transcriptions arrived, we started doing a more systematic analysis of the interviews with NVivo, a qualitative data analysis software. A coding scheme, shown in Figure 1, to code the transcriptions was devised both in a bottom-up fashion, by listening to the interviews and eliciting topics, and top-down fashion by using the main topics included in the research questions/interview script. All transcriptions were coded prior to analysis. The analysis was done by exporting coded parts relating to a certain topic to an Excel sheet where the data could be collected for more easier comparison between UCs.

Generally, we obtained a rich overall picture of the development processes in each UC, and their current practices for managing QRs, as it can be seen from the following sections. However, we plan to conduct more interviews to get further details on those areas identified as particularly relevant for building the Q-Rapids quality aware software development process. Particularly, on those UCs that involve large/complex systems and organizational structure. The results of further interviews will be provided in the next version of this deliverable (M15).
<table>
<thead>
<tr>
<th>Use case company provider</th>
<th>Interviewee profile, P#</th>
<th>Interviewee role (SW = software)</th>
<th>Experience in SW development (Experience in UC provider company)</th>
<th>Experience in agile development</th>
<th>Interview length</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Company 1</strong></td>
<td>P1</td>
<td>Architect, developer, (anything)</td>
<td>10 years (10 years)</td>
<td>10 years</td>
<td>48 minutes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P2</td>
<td>Manage projects that use Product A</td>
<td>18 years (4 years)</td>
<td>10 years</td>
<td>28 minutes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P3</td>
<td>Head of research unit</td>
<td>17 years (11 years)</td>
<td>12 years</td>
<td>48 minutes</td>
</tr>
<tr>
<td><strong>Company 2</strong></td>
<td>P4, P5</td>
<td>DevOps specialist</td>
<td>15 years (2 years)</td>
<td>13 years</td>
<td>54 minutes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Project manager, architect, and product owner</td>
<td>10 years (5 years)</td>
<td>7 years</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P6</td>
<td>Quality manager</td>
<td>21 years (11 years)</td>
<td>9 years</td>
<td>63 minutes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P7</td>
<td>Test automation case</td>
<td>25 years (7 years)</td>
<td>13 years</td>
<td>54 minutes</td>
</tr>
<tr>
<td><strong>Company 3</strong></td>
<td>P8</td>
<td>Deputy requirement area manager</td>
<td>20 years (20 years)</td>
<td>12 years</td>
<td>54 minutes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P9</td>
<td>Local product owner (5G SW) (Project manager of small team)</td>
<td>17 years (4 years)</td>
<td>9 years</td>
<td>41 minutes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P10</td>
<td>Agile coach (LTE SW)</td>
<td>20 years (16 years)</td>
<td>11 years</td>
<td>80 minutes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P11</td>
<td>Project manager</td>
<td>4 years (4 years)</td>
<td>4 years</td>
<td>42 minutes</td>
</tr>
<tr>
<td><strong>Company 4</strong></td>
<td>P12</td>
<td>Product owner, Scrum master, and main software engineer</td>
<td>Almost 9 years (Almost 9 years)</td>
<td>5.5 years</td>
<td>68 minutes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P13</td>
<td>Architect and chief of the SW development department</td>
<td>8 years (8 years)</td>
<td>6 years</td>
<td>47 minutes</td>
</tr>
</tbody>
</table>
Figure 1. Screenshot of coding scheme in NVivo 11 for coding transcribed interviews.

This section presents context information and an aggregated view of the current software development processes followed in the UCs provided as part of the Q-Rapids project (UC1, UC2, UC3 and UC4). Concretely, we present an aggregated view of their general software development processes and, then, we provide details on the current ways of managing QRs in our UCs. Figure 2 gives a picture of the aggregated view that is explained in this section. Individual analysis of each specific UC is provided as an annex (Annex C, Annex D, Annex E, and Annex F, respectively). Because of confidentiality issues these annexes are only privately available for Q-Rapids partners and Commission reviewers. **Once this document is made publicly available, these annexes must be deleted from the document.**

4.1 Context information

The context of the UCs are quite different, which makes it challenging and interesting to compare development processes.

UC1 is a product line producing a modeling tool for developers and architects to support software and system engineering processes. The tool is also used by the company itself. The company has started adopting agile methods in 2006. The team, consisting of 22 persons, does not follow strictly any specified agile process but prefers to utilize a customized agile process. It is based on face-to-face communication, the focus is on working software, the team has close collaboration with customers and users through a sales team and open source community, and they have adopted a way of working that is highly responsive to change. Customers guide the software development process at UC1 and the company usually prioritizes what the user needs.

UC2 develops and maintains highly secure, reliable, and robust connectivity solutions for government and military institutions. The product is an embedded device and is, therefore, also dependent on hardware. The software for the device is developed using agile practices, although there is not the same kind of link to the actual users like in UC1 since the aim is to produce a device for a customer segment to fulfil a market need. UC2 uses a Scrum process in development and maintenance that is well defined.

The provider of UC3 is a global company operating in the telecommunications domain producing embedded systems with both software and hardware. The company has different business lines, of which three are provided as UCs for Q-rapids. All these business lines have several products. Components, which make up a feature of a product, are very big and they are developed by several teams distributed globally at different sites. The different business lines do not all have exactly the same processes. The company is in the transition from a more traditional waterfall like development process to an agile one, however, the transition has started at different times in the different business lines. Business line A is the newest business line that started adoption of agile methods in 2015. Business line B, which is the biggest in number of personnel, started agile adoption in 2011. Business line C started adoption of agile methods roughly at the same time (2009 – 2011).

UC4 is composed of a team of 10 persons that developments software for the health care domain. The relationship between the software development company, customers and users is different in this UC. The customer is in the insurance domain while the users are in the health care domain. The adopted development process is a mixture of Scrum and Kanban. Software is developed in iterative cycles and a Kanban board is used to monitor the status of backlog items. All the software developed by provider of UC4 are special purpose software and are developed in close collaboration with the customer and end-users.

4.2 General software development process

After making an in-depth analysis of the current UCs’ ways of working, we found that all of them base their software development processes in ASD, mainly based on Scrum and Scrum teams. However, each use case
had their own way of interpreting ASD in terms of practices, tools, roles, etc., and were at different maturity levels of agile adoption. For example, whilst one of the UCs was in the state to transition from more traditional waterfall like development to a Scrum like development process, other UCs have been using agile software development for years. In UC3, the process involves many teams with a very complex structure of backlogs. On the other end of the spectrum, UC4 had a much simpler organizational structure and software development process. In this case, they used a mix of Scrum and Kanban and the development process seemed to be well defined so that all involved stakeholders knew the processes and practices to be followed. UC1 provider reported that they do use agile, but not any pre-defined agile practices. In this company, they had defined their own agile way of working with small teams and a lot of face-to-face communication, as little documentation as possible, and utilized frequent communication with users and customers. UC2 provider reported using “Scrum almost directly from the book”.

Commonalities between the UCs could be found. High level features are elicited based on market or customer needs. The level of customer involvement in this process varied amongst the UCs as did the process of how this is done. Some UCs had direct contact with the customers (UC1, UC4) and used mockups when designing solutions for customers (UC4). Other relied on analyzing the market situation and problems of potential customer segments in addition to the involvement of the customer in this early phase (UC2, UC3). These companies aimed more on filling a certain customer segment of the market. There seemed also to be a correlation between customer involvement and size of UC company in the sense that the bigger the company was, the less there was customer involvement. The prioritization of features was also mainly customer demand driven in all of the UCs. The roles responsible for these initial tasks were typically managers of the UCs. There was no differentiation between FRs and QRs in these activities in any of the UCs, but features where managed as complete features including both FRs and QRs.

Similarly, the high-level features were broken down into lower level features or requirements/user stories after the initial elicitation in all UCs, however, there was variability in the number of steps of refining features and also the practices of how this is done. The number of steps also correlated with company size. UC3 had several steps of refinement and specification before actual development, while UC1 and UC4 started refinement and development almost simultaneously. The UCs that had several steps of refinement were typically big organizations and had roles on several levels for refinement, while in smaller organizations the hierarchy was flat and one member of the organization could have several roles, participating in both management activities and development activities. Also in this refinement process, the QRs are managed together with FRs in all of the UCs.

The actual development process was similar for all UCs, they used iterative development and CI. All UCs had predefined release cycles and all UCs, with one exception, had predefined sprints. The release cycles’ length varied from 2 weeks to 6 months. Similarly, sprints were from 1 week to 4 weeks. All UCs used nightly builds and CI was used as the first step of verification and validation in all UCs.

All UCs reported to utilize face-to-face communication, but the amount of face-to-face communication varied greatly. In some UCs (UC1 and UC4) that was the main channel for communication, while in others it was utilized on a minimum level even though developers may be in the same room. In those cases, documentation and tools where the main media for communication. The amount and level of documentation varies similarly. Of those UCs that rely heavily on face-to-face communication, UC1 utilized a minimum amount of documentation. Only issues the UC was not accustomed to were documented. Likewise, those UCs that relied heavily on documentation and tools, used less face-to-face communication.
4.3 Managing quality requirements

Regarding the management of QRs in the UCs, Figure 2 presents an aggregated view of the software development processes at the UCs and their different ways of managing QRS. The bigger circles represent those parts of development where differences in ways of working could be identified. The outer parts of the top right and bottom circles reflect the development process of UCs with complex organizational structure. The inner parts of the circles reflect UCs with less complexity in their development process and practices at lower level in the big UCs. Overall, we observed that QRs are treated together with FRs, i.e., those QRs that makes sense for the UCs are included in, e.g., backlogs as independent items. There is no special way of treating or managing QRs but they flow through the same processes as the FRs. However, it should be noted that FRs get more attention. It was in many cases easier for the UCs to see the value that FRs bring to the customer in comparison to QRs. The UCs follow standards (e.g. ISO 9001), and utilize tools to instill quality in their development process to further ensure quality of software. While they do not have shared or common definition of QRs internally, each of the UCs have QRs specific to their context. In this regard, the UCs adopted different roles, practices, and tools for managing QRs in the requirements elicitation, specification and
analysis, prioritization, documentation, communication, and verification and validation processes. In the following paragraphs, we present QRS management practices adopted in the UCs.

**QRS elicitation:** Involves identifying and clarifying QRS. All UCs prioritize customers as the main source of QRSs. Additionally, organizational roadmaps, standards, sales team, and development team are used as sources of QRSs. QRS elicitation practices involved face-to-face meetings that utilize whiteboards, flip charts and mock-ups. In most of the UCs, roles such as product owner, product manager and technical manager, and sales team are involved in requirements elicitation at higher level. QRS elicitation is done together with FRs in requirements elicitation. At lower levels of development, QRS elicitation is done by development teams during sprints (e.g., elicitation of internally generated QRSs).

**QRS specification and analysis:** Involves specifying and refining QRSs. In some of the UCs, QRSs are specified, analysed and documented. In such cases, requirements management tools, and templates tailored for QRSs, are used for specifying QRSs. Other UCs relied on the minds of developers for analysing QRSs. These developers are experts and they communicate and refine QRSs in face-to-face meetings. They prefer to spend time on development and avoid detailed specification and analysis of QRSs.

**QRS prioritization:** QRS prioritization is done together with FRs at higher levels (in early phase of development). In general, FRs get high priority in cases where QRSs are not specified or are inherited during development. While considering QRSs, determining the highest priority takes into account different factors. In all the UCs, customer issues (QRSs that come from customers) have high priority. However, factors such as security issues, compliance with standards, priority order in backlogs, and blocking issues are also considered in QRSs prioritization decisions. Product and technical managers, and product owners are mainly responsible for QRSs prioritization decisions. Additionally, development teams, architects, and test manager are also responsible in QRSs prioritization decisions. In one of the UCs, the product owner does the prioritization in consultation with development teams. Another UC utilizes the guidance of test manager for QRSs prioritization decisions. At sprint level, QRSs prioritization is done by priority order in backlog in most UCs. However, within a sprint, developers have a say in prioritizing QRSs at least in one of the UCs. In general, FRs get usually higher priority compared to QRSs.

**Communication of QRSs:** Communication plays a great role in clarifying QRS both with customers and within development teams. Face-to-face meetings are mainly utilized to clarify QRS during daily stand up meetings, and weekly and biweekly meetings in the UCs. However, the timing, and roles involved in these meetings varies in each of the UCs. Additionally, documentation in project wiki and shared resources and tools, sales team, hotlines for customers, issue trackers, forums, and release notes are other means of communicating QRS within development teams and among customers.

**Documentation of QRSs:** Regarding the documentation practices, Word documents, Power points, spreadsheets, mock-ups, backlogs at different levels, and wiki pages are used to document QRSs. Additionally, QRSs are documented in acceptance criteria and Definition of Done (DoD). However, in one UC, QRSs were not documented. In this case, QRSs were being communicated in face-to-face meetings facilitated by white boards and flip charts.

**Verification and validation of QRSs:** The UCs applied different practices for verifying and validating QRSs. QRS are included in the acceptance criteria and the DoD at different levels (e.g. user stories, tasks, and tickets). Additionally, the UCs apply code review, testing at different levels (unit, component, and system), CI, check lists for checking QRS at different levels, and manual testing for verifying and validating QRSs. Roles involved in the verification and validation process included development team, testers, open source community, validation teams, product owners, and quality assurance engineers.
In general, the UCs strive for ensuring quality of their product and processes, and do not differentiate between QRs and FRs.

4.4 Challenges and areas of improvement

This section lists the different challenges and areas of improvement identified from the interviews. Not all challenges are reported as challenges just for QRs but also for FRs. This could be expected since companies reported to manage QRs together with FRs.

Challenges identified during our interviews included:

- **QRs are not documented properly**: In some of the UCs, QRs are often communicated in face-to-face meetings through whiteboards or flip charts and very often end up staying in the minds of the managers and developers. This creates a problem for new developers that join the teams, as there will not be traceability of QRs. Additionally, developers focused more on delivering value and saw documentation as slowing the development process. As a result, QRs were not documented properly.

- **Requirements understood differently due to communication issues**: In two UCs, we found that often QRs are understood differently between customers and development team due to communication issues. Additionally, in some UCs, communication of requirements (both FRs and QRs) within development team and among different teams is also challenging. This occurs when developers completely rely on auto generated e-mail communication or due to organizational structure.

- **Too late consideration of QRs**: One interviewee emphasized that developers want to bring in QRs and their thresholds in early stages of the software development process. Late consideration of QRs was seen as a challenge.

- **Difficult to see what brings value**: Identifying the value behind specific QRs is one of the challenges shared among different UCs. Developers and decision makers are interested in determining and showing the value of QRs.

- **Value based QRs prioritization**: When QRs are not prioritized by taking into account their business value, they have consequences in the development process (e.g. cost, development of unnecessary features). Identifying and prioritizing QRs that bring value is a common challenge in two UCs. In such cases, companies are not able to easily identify QRs that bring value and may spend time developing unnecessary features.

- **Limited or no visibility to original requirements**: In some of the use cases, the interviewees have limited visibility of the higher-level requirements due to their large and complex organizational structures. Teams work on requirements that are fed into their backlogs but without visibility on the higher level reasons why that requirement is being implemented.

- **Internally inherited QRs are seldom prioritized**: Internally inherited QRs such as operability arise during the software development process. One of our interviewees stated that such types of QRs are seldom prioritized. Instead, there was much emphasis on delivering value and thus QRs that are explicitly defined in the backlogs were prioritized. This challenge is, therefore, related to the difficulties to determine the value of QRs.

- **Difficulty in showing QRs in backlogs**: Interviewees in one UC identified requirements management tools as problematic and had problems in showing QRs in related backlogs.

- **How to get the most information in initial scoping session**: Our interviewees identified that product owner’s and development team’s limited knowledge of a specific domain area contributes to challenges of QRs in ASD. Additionally, developers have challenges in foreseeing and responding to QRs that customers are not able to see in early stages of development. The lack of strategies and soft skills for collecting a comprehensive list of QRs are the reasons behind this challenge.
We find that some challenges of QRs reported in the UCs, such as late consideration of QRs, and problems with documentation of QRs (frequent reliance on tacit knowledge of developers) are also reported in the scientific literature. Additionally, the challenges identified in the interview analysis (e.g. visibility of QRs during product lifecycle) align with the challenges of QRs management that were pointed out in the Q-Rapids’ proposal.

Similarly, the UCs are interested in the following areas of improvement:

- Ability to get real time feedback on usage data from customers to learn about feature usage.
- Developers want to understand the motivation behind roadmap decisions that are made by top management.
- Achieving real-time feedback, value stream visualization of QRs, and earlier feedback on QRs in the development process.
- Improved visibility of QRs during the development process.
- Understanding the impact of code change on other requirements (QRs and FRs).
- Improvement in the estimation of tasks implementation, developing the soft skills in requirements gathering, generalizing the reusability of components.
5. Implications for the Q-Rapids framework

The findings from the state-of-the-practice of QRs management in the UCs serve as a baseline for defining a process baseline of the Q-Rapids framework. Different roles, practices, tools and methods, and activities are determined while examining the “As-is” practice of managing QRs in the UCs. Current software development process, challenges in the QRs management practices and areas of improvement have been identified in each of the UCs.

All the UCs involved in Q-Rapids follow an agile-based software development process. However, as it is natural, different UCs have different particularities in their way of implementing ASD (customized agile processes based on their own particular contexts and on their different adoption levels of ASD and RSD). In addition, the contexts of the UCs are different and the development organizations look different as well. One UC may be composed by a single development team, while another one may have multiple larger teams. The development process in these large organizations is not as simple and straightforward as in the smaller once, but is more complex.

For example, some of the UCs have restricted access to end-user system usage due to confidentiality reasons, and cannot for that reason utilize this part of the Q-Rapids framework (Figure 3) to its full extent. The utilization and feeding of project and development data to the quality-aware software development process will most likely need to be implemented at different levels in different use cases. An implementation satisfying a small UC with small development team will most likely not satisfy the needs of a large UC with multiple development teams and complex dependencies between teams and business lines. The same can be said about the exchange of requirements between the backlog, or many different backlogs on many levels. For these reasons, the Q-Rapids framework needs to be flexible and able to be adapted to the UCs own development processes based on their own context and needs.

The findings reveal also the opportunities that will be addressed in the Q-Rapids framework and serve as an initial step in determining process baseline metrics. The findings of this deliverable serve as an input for WP1 (data gathering and analysis) and WP3 (strategic decision making dashboard). For example, process related terms (see Annex G – Process related terms) including roles and activities identified in the use cases are
shared with WP3. Similarly, information of tools used in the software development process of the use cases identified in the analysis is shared with WP1.
Conclusions

The deliverable presents an overview of the state-of-the-art and the state-of-the-practice of QRs management practices. Related to the state-of-the-practice, our results are based on an in-depth analysis of current ways of working and QRs management practices in the Q-Rapids’ UCs. Our findings reveal that QRs are managed together with FRs in all of the UCs regardless of the difference in context of the UCs. The UCs focus in ensuring quality of the product and development process in general. In some of the UCs specific QRs were documented, while other UCs relied on just communicating QRs inside the development teams. Identifying the value of QRs and their impact was an interest for interviewees. Areas of improvement envisioned by the UCs included improved visibility of QRs in the development process, achieving value stream visualization of QRs and earlier feedback on QRs, and increasing reusability of components.

In this deliverable, we presented a descriptive analysis of the related literature. In addition, we are conducting a SLR on the topic to be published as a journal article. The findings on the state-of-the-art will be used to complement our finding on the state-of-the-practice of QRs management in the UCs, as an input for driving quality-aware RSD process.

This deliverable will be reviewed during the project lifecycle. New information that emerge during the project will be incorporated to complement our current understanding on processes for managing QRs in the UCs.
References


Annex A – SLR protocol

To analyse the scientific literature on managing QRs in the context of ASD and RSD, we are conducting a systematic literature review (SLR) in the topic according to the guidelines by Kitchenham et al. [1]. We have developed a protocol to guide the execution of the SLR, which is being conducted as a collaborative activity between research partners in the Q-Rapids consortium. The protocol includes the typical steps described by Kitchenham et al. [1] for conducting SLRs: identifying the research objective, planning and executing the search strategy, study selection and quality assessment, data extraction, and data analysis and synthesis. The activity is currently on-going. Next, we describe each step of the process and detail the current status.

1. Identifying the research objective: The objective of the SLR is to synthesize the state of the art on managing QRs in ASD and RSD by examining relevant scientific literature. We decided to conduct this SLR because, despite the existence of related secondary studies on agile requirements engineering ([2], [3], [4], [5], and [6]), none of these studies had a particular focus on QRs. The results of the SLR will be used as an input for developing our Q-Rapids software development process.

2. Planning and executing the search strategy: To decide the search strategy to be followed, we experimented with both automatic database search (run on Scopus, Web of Science, ACM Digital library, and Science Direct) and snowballing (using primary studies of secondary studies in agile requirements engineering and RSD). The automatic search created a lot of noise (returned many false positive studies) and did not retrieve as many relevant papers as the snowballing strategy. Therefore, we decided to use snowballing.

3. Study selection: Inclusion/exclusion criteria to do the study selection have been defined as follows:

   Inclusion criteria:
   - Publications that contribute to the body of knowledge of QRs in ASD or RSD (e.g. scientific papers that discuss different aspects of QRs in the context of ASD or RSD).
   - Scientific papers including book chapters (i.e. papers published in scientific workshops, conferences, or journals)
   - Peer reviewed publications
   - Publications only written in the English language

   Exclusion criteria:
   - Publications that do not contribute to the body of knowledge of QRs in ASD or RSD (e.g. Scientific papers that only mention or discuss QRs in a context other than ASD or RSD; papers that only mention specific practices of ASD or RSD without further discussion of QRs)
   - Books, blogs, short papers, tutorials etc.
   - Secondary studies (SLRs and systematic mapping studies)
   - Exact duplicates of a study
   - Non-English publication

   Six researchers piloted the inclusion/exclusion criteria on 36 studies identified from seven related secondary studies ([2], [3], [4], [6], [7], [8], [9]). We are currently conducting the search process and have included 14 primary studies so far.

4. Quality assessment: The quality of selected primary studies will be systematically assessed in order to ensure that only results from quality papers are included in the SLR. The criteria used for assessing the primary studies’ quality will be piloted among six researchers.
5. **Data extraction**: The following data properties will be extracted in the data extraction process. Additionally, detailed data extraction forms are used to guide the process and extract data to address our research objective.

<table>
<thead>
<tr>
<th>Data ID</th>
<th>Data property</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1.1</td>
<td>Publication year</td>
</tr>
<tr>
<td>D1.2</td>
<td>Publication venue type</td>
</tr>
<tr>
<td>D1.3</td>
<td>Research method</td>
</tr>
<tr>
<td>D1.4</td>
<td>Research rigor</td>
</tr>
<tr>
<td>D1.5</td>
<td>Industrial relevance</td>
</tr>
<tr>
<td>D2.1</td>
<td>Domain</td>
</tr>
<tr>
<td>D2.2</td>
<td>Type of QRs</td>
</tr>
<tr>
<td>D2.3</td>
<td>QRs definition</td>
</tr>
<tr>
<td>D2.4</td>
<td>How the reported QRs is measured</td>
</tr>
<tr>
<td>D3.1</td>
<td>Strategy to manage QRs (practice, tool, model, method)</td>
</tr>
<tr>
<td>D3.2</td>
<td>Reported impact of strategy</td>
</tr>
<tr>
<td>D4.1</td>
<td>Base measure for Key Strategic Indicator (KSI)</td>
</tr>
<tr>
<td>D4.2</td>
<td>Real data source for metric/measurement</td>
</tr>
<tr>
<td>D4.3</td>
<td>KSI of QRs</td>
</tr>
<tr>
<td>D4.4</td>
<td>How KSI relate to the QRs</td>
</tr>
<tr>
<td>D4.5</td>
<td>Tool to manage KSI</td>
</tr>
<tr>
<td>D5</td>
<td>Challenges relating to QRs in ASD or RSD</td>
</tr>
</tbody>
</table>

**Data analysis and synthesis**: We will follow a systematic data analysis and synthesis process based on Cruzes and Dybå [7] thematic synthesis guidelines. NOTE: References can be found in the reference list of the deliverable.
Annex B – Interview script

1. WARM-UP QUESTIONS:

Q1.1: **Tell us something about your work experience?** [Subject’s experience may impact his/her perspective on understanding of quality requirements management in rapid software development]

   ✓ How long have you been working in the company?
   ✓ How long have you been working with the product line?
   ✓ What is your role in the development/project?
   ✓ Do you have any previous experience in software development? If yes how long and in what roles?
     [Experience in traditional, agile and rapid software development before working in this company]
     o How many years of experience in agile and rapid software development?
     o In what roles?

Q1.2: **Tell us something about the product you are working on?** [The type of product may impact quality requirements management approach employed in rapid software development]

   ✓ What is the name of the product?
   ✓ What is the domain area of the product? (E.g. financial services, telecom, manufacturing industry, etc.)
   ✓ What is the type of the product? (e.g. proprietary or open-source)
   ✓ What is the size of the product? (e.g. in LOCs)
   ✓ What is the programming language used for developing the product?
   ✓ Is the product general or customized to different customer segments?

2. OVERALL DEVELOPMENT PROCESS WITH EMPHASIS ON QRs MANAGEMENT QUESTIONS:

Q2.1 How would you describe your way of working? [Do you use waterfall, agile, scrum, hybrid, etc?]

   ✓ Could you please give us brief description of the practices and activities that you have or the different processes that you follow from requirements definition to validation?

Q2.2 How do you understand quality requirements (in your company)?

   ✓ Do you have a shared and established definition/conceptualization of quality requirements?
   ✓ What are the constituents/elements of quality requirements typical to your development?

Q2.3 In your opinion, are quality requirements important for the development/product?

   ✓ If yes, what are the most important quality requirements in your development?
   ✓ If no, why?

Q2.4 How are quality requirements taken into consideration during development?

   ✓ Do you treat quality requirements and functional requirements at the same time or separately?
   ✓ What are the sources of quality requirements in the context of your development?
     o Do you have internal sources (e.g. development teams, customer representatives, maintenance teams), external sources (e.g. customers, end users of the product) for quality requirements?
     o Do you have standards, rules, and laws you need to abide by that brings quality requirements? [e.g. security standards and policies that may affect quality requirements]
   ✓ How do you identify and elicit functional requirements?
     o Who is responsible for functional requirements elicitation?
     o What kind of tools, practices, and techniques do you use for eliciting functional requirements?
     o What are the current challenges in the identification and elicitation process of functional requirements?
   ✓ How do you identify and elicit quality requirements?
○ Who is responsible for quality requirements elicitation?
○ What kind of tools, practices, and techniques do you use for eliciting quality requirements?
○ What are the current challenges in the identification and elicitation process of quality requirements?
○ What areas of improvement would you like to see in the requirements elicitation process?

✓ How do you communicate quality requirements with customers?
○ What kind of tools, practices, and techniques do you use for communicating quality requirements?
○ What are the challenges of communicating quality requirements with customers?
○ What areas of improvement would you like to see while communicating quality requirements with customers?

✓ How do you communicate quality requirements internally? [Within teams and with other teams?]
○ What kind of tools, practices, and techniques do you use for communicating quality requirements internally?
○ What are the challenges of communicating quality requirements internally?
○ What areas of improvement would you like to see while communicating quality requirements internally?

✓ How do you document quality requirements? [If yes, how?]
○ What kind of tools, practices, and techniques do you use for documenting quality requirements?
○ What are the current challenges regarding documentation of quality requirements?
○ What areas of improvement would you like to see in the documentation of quality requirements?

✓ How do you do requirements analysis?
○ Do you specify or model requirements?[both functional and quality requirements]
○ If yes, do you specify or model functional requirements together with quality requirements?
  ▪ How do you specify or model them?
○ If functional requirements and quality requirements are treated separately:
  ▪ How do you specify or model functional requirements?
  ▪ How do you specify or model quality requirements
○ What kind of tools, practices, and techniques do you use in the requirements analysis?
○ What are the current challenges regarding specification of requirements? [Both functional and quality requirements]
○ What areas of improvement would you like to see in the requirements analysis process?

✓ How do you prioritize quality requirements in your development?
○ Are there any rules for prioritizing quality requirements?
○ When and how often are they prioritised?
○ Who does the prioritisation / is responsible for prioritisation?
○ How do you communicate information regarding quality requirements prioritization with relevant stakeholders?
○ What are the current challenges regarding prioritisation of quality requirements?
○ What areas of improvement would you like to see in the quality requirement prioritization?

✓ How do you validate the quality requirements?
○ What are the techniques you use for validating quality requirements?
○ If review meetings, how often and who are present?
○ Who determines if quality requirements have been met sufficiently?
○ How is user feedback regarding QRs taken into consideration during development?

Q2.5 How does requirements engineering (requirements development and management process), and in particular quality requirements, relate to other areas?

✓ Are quality requirements taken into considerations in the risk management process?
○ Does the exclusion of quality requirements increase risk in the project? [If yes, in what ways?]
○ Does risk management introduce new quality requirements?
✓ How are quality requirements affected in project planning?
Do quality requirements affect project planning? [If yes, how?]
- Are quality requirements taken into account during project planning?
  - Are quality requirements budgeted in project planning?
  - How does the availability of persons/resources management affect quality requirement?
- How are quality requirements related to project management?
  - Does management see the importance of quality requirements?
  - Are quality requirements taken into consideration during budgeting?
  - How are issues regarding quality requirements communicated between development and project management?

3. WRAP-UP QUESTIONS:

Q3.1 Do you see any challenges in the current methods, techniques, tools, or practices regarding quality requirements management?
- What areas of improvement would you like to see related to QR management?
- Do you have any suggestions for improvement?

Q3.2: Are there any related issues that we missed and that you would like to reflect on?
Annex C – UC1
Annex removed from the publically available version of this deliverable because of confidentiality restrictions.
Annex D – UC2

Annex removed from the publically available version of this deliverable because of confidentiality restrictions.
Annex E – UC3

Annex removed from the publically available version of this deliverable because of confidentiality restrictions.
Annex F – UC4

Annex removed from the publically available version of this deliverable because of confidentiality restrictions.
Annex G – Process related terms

Acceptance criteria
Description: specify the values that a product must meet for acceptance (e.g., a performance requirement that a function must be executed within one second).

Source(s): [Wallace & Cherniavsky: 1990]

Agile Development
Description: software development approach based on iterative development, frequent inspection and adaptation, and incremental deliveries, in which requirements and solutions evolve through collaboration in cross-functional teams and through continuous stakeholder feedback.


Developer
Description: role responsible for writing the code of the story.

Source(s): [Leffingwell, D. 2011]

Epic
Description: Highest level expression of a customer or business need.

Source(s): [Leffingwell, D. 2011]

Feature
Description: functional or non-functional distinguishing characteristic of a system, usually an enhancement to an existing system.


Feature Team
Description: long lived cross functional team that completes many end to end customer features, one by one.

Source(s): [Leffingwell, D. 2011]

Process
Description: a set of interrelated or interacting activities which transforms inputs into outputs.

Source(s): [ISO 9000:2005]

Product Manager
Description: person who is responsible for defining the features of the system at program level.

Source(s): [Leffingwell, D. 2011]

Product Owner
Description: is responsible for determining and prioritizing user requirements, managing the product backlog.

Source(s): [Leffingwell, D. 2011]

Project Portfolio
Description: collection of projects that addresses the strategic objectives of the organization.

Source(s): [ISO 12207: 2008]
Quality

Description: degree to which a set of inherent characteristics fulfils requirements.

Source(s): [ISO 9000: 2005]

Rapid Software Development

Description: evolutionary step from agile software development that focuses on organizational capability to develop, release, and learn from software in rapid parallel cycles, such as hours, days or very few weeks’

Source(s): [Mäntylä, Adams, Khomh, Engström & Petersen: 2015] and [Fitzgerald, B & Stol, 2017]

Release

Description: describes an increment that is valuable to customers and evolves into complete software product

Source(s): [Greer & Ruhe: 2004]

Requirement

Description: need or expectation that is stated, generally implied or obligatory.

Source(s): [ISO 9000: 2005]

Requirements prioritization

Description: Crucial and integral part of software decision making that helps to identify the most valuable requirements among candidate requirements that need to be realized within time and cost constraints.

Source(s): [Berander, P & Andrews: 2005]

Sprint

Description: short time frame, in which a set of software features is developed, leading to a working product that can be demonstrated to stakeholders. NOTE In some organizations, a sprint is known as an iteration.


Task

Description: are individual work items which compose a requirement, are commonly assigned to individuals, and in many cases are formally represented in issue trackers and code repositories.

Source(s): [Ernst et al.:2012]

Team backlog

Description: typically called project or product backlog consists of all the user stories the team has identified for implementation.

Source(s): [Leffingwell, D. 2011]

Tester

Description: integral part of agile team responsible for writing acceptance test case while the code is written and testing it against the acceptance criteria.

Source(s): [Leffingwell, D. 2011]

User Story

Description: simple narrative illustrating the user goals that a software function will satisfy.

References used for process related terms


