Quality-Impact Assessment of Software Products and Services in a Future Internet Platform

Farnaz Fotrousi

Blekinge Institute of Technology Licentiate Dissertation Series
No 2015:09

Quality-Impact Assessment of Software Products and Services in a Future Internet Platform

Farnaz Fotrousi

Licentiate Dissertation in
Telecommunication Systems

Department of Communication Systems
Blekinge Institute of Technology
SWEDEN

2015 Farnaz Fotrousi
Department of Communication Systems
Publisher: Blekinge Institute of Technology,
SE-371 79 Karlskrona, Sweden
Printed by Lenanders Grafiska, Kalmar, 2015
ISBN: 978-91-7295-318-5
ISSN 1650-2140
urn:nbn:se:bth-10949

Abstract

The idea of a Future Internet platform is to deliver reusable and common functionalities to facilitate making wide ranges of software products and services. The Future Internet platform, introduced by the Future Internet Public Private Partnership (FI-PPP) project, makes the common functionalities available through so-called Enablers to be instantly integrated into software products and services with less cost and complexity rather than a development from scratch.

Quality assessment of software products and services and gaining insights into whether the quality fulfills users' expectations within the platform are challenging. The challenges are due to the propagation of quality in the heterogeneous composite software that uses Enablers and infrastructure developed by third parties. The practical problem is how to assess the quality of such composite software as well as the impacts of the quality on users' Quality of Experience (QoE).

The research objective is to study an analytics-driven Quality-Impact approach identifying how software quality analytics together with their impact on QoE of users can be used for the assessment of software products and services in a Future Internet platform.

The research was conducted with one systematic mapping study, two solution proposals, and one empirical study. The systematic mapping study is contributed to produce a map overviewing important analytics for managing a software ecosystem. The thesis also proposes a solution to introduce a holistic software-human analytics approach in a Future Internet platform. As the core of the solution, it proposes a Quality-Impact inquiry approach exemplified with a real practice. In the early validation of the proposals, a mixed qualitative-quantitative empirical research is conducted with the aim of designing a tool for the inquiry of user feedback. This research studies the effect of the instrumented feedback tool on QoE of a software product.

The findings of the licentiate thesis show that satisfaction, performance, and freedom from risks analytics are important groups of analytics for assessing software products and services. The proposed holistic solution takes up the results by describing how to measure the analytics and how to assess them practically using a composition model during the lifecycle of products and services in a Future Internet platform. As the core of the holistic approach, the Quality-Impact assessment approach could elicit relationships between software quality and impacts of the quality on stakeholders. Moreover, the early validation of the Quality-Impact approach parameterized suitable characteristics of a feedback tool. We found that disturbing feedback tools have negligible impacts on the perceived QoE of software products.

The Quality-Impact approach is helpful to acquire insight into the success of software products and services contributing to the health and sustainability of the platform. This approach was adopted as a part of the validation of FI-PPP project. Future works will address the validation of the Quality-Impact approach in the FI-PPP or other real practices.

Keywords: Quality of Experience (QoE), Quality-Impact, analytics, software quality, KPI, Future Internet, Quality of Service (QoS), assessment

Acknowledgments

I would like to express my sincere appreciation to my supervisors Prof. Dr. Markus Fiedler and Prof. Dr. Samuel A. Fricker for their continuous, invaluable, and helpful support and guidance of my research. Despite their busy schedules, they were always available to share their deep knowledge and give me their insightful feedback in my study.

Thanks to my colleagues in DIKO and DIPT departments for the enjoyable and educating conversations we have had. Especially, I would like to thank Deepika and Indira for their supportive discussions and their friendships.

I also use this opportunity to express my gratitude to partners involved in the FI-STAR project, collaborated kindly and provided me with the opportunity of doing these studies.

My deep appreciation goes to my family for being so encouraging and supportive. Especially, I am so grateful to my spouse, Shahryar for his great emotional and technical support along the way. He was truly my invaluable consultant, who continuously shared with me his knowledge and insight.

At last, a special gratitude to my father, for the life-long devotion, love, and support he bestowed on me. He will always take part in the bests that I will achieve through my life.

Overview of Papers and Deliverables

Papers in this Thesis

Chapter 2: Farnaz Fotrousi, Samuel Fricker, Markus Fiedler, and Frank Le-Gall. "KPIs for software ecosystems: A systematic mapping study." *5th International Conference on the Software Business (ICSOB 2014),* Paphos, Cyprus, 2014.

Chapter 3: Farnaz Fotrousi, Samuel Fricker, and Markus Fiedler. "Quality Requirements Elicitation based on Inquiry of Quality-Impact Relationships. " *IEEE 22nd International Conference on Requirements Engineering (RE), Karlskrona, Sweden,* 2014.

Chapter 4: Samuel Fricker, Farnaz Fotrousi, and Markus Fiedler. "Quality of Experience Assessment based on Analytics." *2nd European Teletraffic workshop* (ETS 2013), Karlskrona, Sweden, 2013.

Chapter 5: Farnaz Fotrousi, Samuel Fricker, and Markus Fiedler. "The effect of requests for user feedback on Quality of Experience." To be submitted to *Software Quality Journal*, 2015.

Related Papers

Paper 1: Farnaz Fotrousi, Katayoun Izadyan, and Samuel Fricker. "Analytics for Product Planning: In-depth Interview Study with SaaS Product Managers." *Sixth International IEEE Conference on Cloud Computing (CLOUD)*, Santa Clara Marriott, CA, USA, 2013.

Overview: The paper empirically identifies the important analytics for product planning decisions in service-based software products, and describes the strengths and limitations of using analytics to support product managers' decisions.

Paper 2: Samuel Fricker, Kurt Schneider, Farnaz Fotrousi, Christoph Thuemmler. “Workshop videos for requirements communication.” Requirements engineering, pp 1-32, 2015. doi: 10.1007/s00766-015-0231-5

Overview: The paper presents a workshop video technique and a phenomenological evaluation of its use for requirements communication. The technique elicits and uses the feedback from the users who provide their perceptions with a video’s experience by annotating the video and expressing the rationale behind each annotation.

Related Deliverables

Deliverable 1: “D6.2 Common test platform.” FI-STAR Public Deliverables, June 2014.

Overview: The deliverable describes the common test platform including the methods and tools for measuring the key performance indicators (KPIs) defined within the European FI-STAR project. My contribution was to propose the test platform for Quality of Experience (QoE) and Quality of Service (QoS) KPIs provisioning and describe how data will be collected and analyzed.

Deliverable 2: “D6.4 Validated Services at Experimentation Sites.” FI-STAR Public Deliverables, October 2015.

Overview: The deliverable represents the validation of FI-STAR applications at the experimentation sites. My contribution was to measure, validate and report the impact of the FIWARE platform as a Future Internet platform on the End-to-End QoE & QoS of FI-STAR applications.

Contents

List of Tables

List of Figures

CHAPTER 1 Introduction

1.1 Overview

The recent years have observed an extensive development of software products and services in Future Internet. Future Internet expects a huge number of new interesting software and mission-critical services that require advanced levels of interoperability (Zahariadis et al., 2011) between software components, products or services. Also, interconnectivity between software and devices as well as self-configuring capabilities within the Internet infrastructure are some of the features that characterize Future Internet.

Future Internet is an ongoing research paradigm, investigated through many research projects. The FI-WARE project as a project of the European Future Internet Public-Private-Partnership (FI-PPP) program aims at an open platform to facilitate the creation of software with less cost and complexity to serve users at large scale based on the Internet (FI-WARE, 2015). FI-WARE delivers a service platform including reusable and shared functionality components. The components are referred as Enablers whether developed for a general purpose or for a specific domain purpose such as health (FI-STAR, 2015). An Enabler makes its functionality available to software products and services through APIs. This is usually an easier and cheaper alternative to the development of the functionality from scratch in software products and services. The functionality of the Enabler is served by a Future Internet infrastructure where the Enabler may be deployed on a different node (e.g. virtual machine) than the software node.

The owner of the platform has to ensure that the platform is healthy (Costanza & Mageau, 1999) and sustainable (Chapin, Torn, & Tateno, 1996). The platform owner monitors the health of the platform in terms of its productivity for the surrounding stakeholders in order to identify the risk factors that threaten maintainability of the platform and shorten the time of recovery (Costanza & Mageau, 1999). One of the risk factors is about the quality of Enablers especially when they are integrated and used in software products and services. Even though an Enabler may pass the internal quality checking, it may be unsuccessful to provide a good level of quality in use (Bevan, 1999) in integration with the software to fulfill the users' expectations.

Lack of insight into the quality of a platform and the impact of quality on users makes the platform owner unable to assure that user demands have been satisfied with the platform. Dissatisfaction of end users from software products and services caused by quality shortcomings of the platform endangers the health of the platform. This problem increases the likelihood of user churn, meaning that the users discontinue using the software products or services. In consequence, the product owners are discouraged to continue using the platform services, which negatively influences the platform's sustainability. To reduce the likelihood of losing customers and improving the platform success, a software-human analytics-driven approach for assessment of software products and services in a platform can be a solution.

For the platform owners who need to monitor the success of the platform, using a Quality-Impact assessment approach is a suitable solution. The approach measures the quality of software products and services that use the platform services together with the impact of such quality on users. The approach uses software-human analytics relevant to the quality of software and platform as well as to the Quality of Experience (QoE) of the users during an experience of the software and platform. The approach also combines the quantitative analysis with qualitative feedback to interpret the analytics. Unlike just pure software quality analytics or pure QoE analytics or pure qualitative feedback, the combination of the two categories of analytics and qualitative feedback allows understanding the level of quality that could keep users satisfied. And, if they are not satisfied, the owners can perform a root cause analysis to identify whether the causes are shortcomings of the software quality, quality issues with Enablers, or even the quality of the underlying communication networks.

Software quality analytics and the impacts of the quality on users' perception together are important for assessment. Software quality analytics gives insights into the quality of software, integrated Enablers in the software or the involved networks, but it does not say anything about users' satisfaction with the quality. Human quality analytics reflects the Quality of Experience (QoE) of users in terms of the "degree of delight or annoyance of the user" (Le Callet, Möller, & Perkis, 2012) with the software, but does not reveal whether the quality of the software, the quality of the integrated Enablers, or some other factors (Reiter et al., 2014) have influenced on a bad experience. Therefore, software quality analytics and human analytics together complement each other for assessment. Moreover, analytics cannot replace or be replaced by qualitative user feedback (Fotrousi, Izadyan, & Fricker, 2013). Analytics is not able to identify why users are not satisfied (Clifton, 2012) but qualitative feedback is. Qualitative user feedback is not able to provide insight into software quality, but analytics is. Qualitative feedback interprets analytics and fills up the gap between the users' perceived quality and software quality.

This licentiate thesis provides an overview of literature to understand main objectives of platform owners as well as the analytics that they use for managing a software ecosystem. The thesis takes the findings and proposes two complementary assessment approaches based on using software-human analytics. As the early validation of the solutions, the thesis investigates the characteristics of a supportive tool for the proposed approaches.

The first study is a systematic mapping research to give an overview of literature that addresses Key Performance Indicators (KPI) in a software ecosystem. KPIs are those among the many possible analytics that are important, easily measurable defined based on the platform owner' objectives. The study provides classification maps to overview the KPIs that are commonly used by platform owners.

The second and third studies describe approaches for assessments of software products and services. The second study describes a Quality-Impact inquiry approach. The study models the relationship between software quality and its impact on users during a software experience. The third study describes the composition of software-human analytics for assessment of software products and services in the Future Internet platform.

The last study is conducted in an empirical research as an early validation of the proposals. It aims at understanding the effect of feedback tool on Quality of Experience of the software that the feedback is collected for. It also parameterizes characteristics of a feedback tool to be used for designing a proper tool suitable for inquiries of feedback.

The thesis is structured in five chapters as follows. Chapter 1 provides an introduction to the licentiate thesis. The rest of this chapter gives an overview of the related areas to the thesis in Section 1.2. Sections 1.3 and 1.4 present the thesis objectives and research questions respectively. Section 1.5 overviews the research methods used to address the research questions as well as threats to the validity of the research. Section1.6 presents the summary of results and overviews the included studies in the thesis. Section 1.7 discusses the author's contributions. Section 1.8 concludes the thesis document with the future works. Each of the next four Chapters, 2, 3, 4 and 5 addresses one research paper included in the licentiate thesis.

1.2 Background

A platform owner needs to have insight into how the software products and services, which are utilizing the platform' services, make their users satisfied to return using again. Acquiring such insight requires a combination of analytics relevant to software quality and impact of the quality on users. This section starts with the concept of analytics, how to acquire the analytics and basic definitions used in this concept. Then software quality and relevant standard models will be described. At last, the Quality of Experience (QoE) and its measurement models will be explained.

1.2.1 Analytics

Analytics is a source of information to guide managers in their decisions. It is known as the data-centric style of decision making (Buse & Zimmermann, 2010) that includes measurements to generate data and to transform these data into indicators for decision support. In another word, analytics is a use of statistics from measurement characteristics of an entity (Davenport & Harris, 2007) to obtain insight and actionable information (D. Zhang et al., 2011) and to take data-driven decisions (Buse & Zimmermann, 2010, 2012).

Three types of analytics can be used for decision-making: descriptive, predictive, and perspective analytics (Delen & Demirkan, 2013). Descriptive analytics summarizes available data to inform decision makers of what happened or is happening. Predictive analytics uses historical data to detect data patterns and forecast a stimulus relevant to software in the future to answer what will happen. Perspective analytics, as a type of predictive analytics, also includes actionable data and feedback to track the data to propose the best course of actions for a given objective. This type of analytics uses complex mathematical algorithms with techniques such as optimization modeling, expert system and multi-criteria decision making (Delen & Demirkan, 2013).

Analytics is made through a chain of interrelated activities. The activities introduce keywords in the Italic style that will be defined:

1- Measuring a set of *measurement attributes* of an *entity* through a *measurement function* to build *metrics*

2- *Analyzing* the *metrics* in the context of an analysis model to have *indicators*

3- *Interpreting* the *indicators* to make required information for a decision-maker

The steps define the measurement information model ISO/IEC 15939 (ISO/IEC-15939, 2007) for software development and systems engineering. The involved keywords in a measurement chain can be defined as:

- *Entity* is a platform, product, feature or requirement considered relevant to the interaction between an end user and a software product, service, or platform (e.g. product).

- *Measurement attributes* are properties or characteristics of an entity relevant to information needs that can be distinguished quantitatively (e.g. returning users).

- A *Measurement method* is a logical sequence of operations that quantify a *measurement attribute* numerically by mapping it to a scale. One or more measurement attributes can be the input for a measurement method.

- A *Measure* is a variable that a value is assigned to, as the result of the measurement. (e.g. number of returning users for product x in recent month).

- *Analysis* is an algorithm or calculation that combines measures by considering decision criteria. Model is an alternative terminology for the analysis. The analysis is usually performed based on the expected relationship between measures and/or their behaviors over time.

- *Indicator* provides an estimate or evaluation of specified measurement attributes derived from a model of defined information for decision-making (e.g. churn rate).

- *Interpretation* explains the quantitative information in the *indicators* to the information needs in the language of the decision maker (e.g. new product release decision).

Quality-related analytics is the kind of analytics that considers quality attributes to be measured. The quality attributes are relevant to the quality of software (software analytics) or impact of the using the software on stakeholders (human analytics). Quality-related analytics excludes the rest of analytics such as business and financial analytics. The next two sections will describe the two categories of quality characteristics.

1.2.2 Quality of Software

Software quality measures how well software is designed, implemented and conforms to users' requirements and standards. Several studies model the software quality (Klas, Heidrich, Munch, & Trendowicz,

2009) with the objective of providing a framework for the evaluation of software quality. ISO/IEC-9126 (ISO/IEC-9126, 2001[part1] - 2003 [part2, part3]) is one of the most popular standards that models the quality in a form of a taxonomy of quality characteristics and sub-characteristics. The standard also defines a set of external, internal and quality-in-use metrics for measurement of the characteristics and sub-characteristics. Internal metrics measure the software itself in the static mode and do not rely on software execution. The external metrics measure the behavior of running software. Quality-in-use metrics measure the impact of using software on stakeholders when users experience the software in a real specific context of use.

ISO/IEC 25010 (ISO/IEC-25010, 2010) evolves the ISO/IEC 9126 with few changes in the taxonomy of quality characteristics and sub-characteristics but on the same basis. The model describes quality from perspectives of product quality and quality-in-use. ISO/IEC 25010 defines the same six software quality categories of characteristics (i.e. functionality, reliability, usability, efficiency, maintainability, and portability) as well two more categories (i.e. security and compatibility). The characteristics are broken down into sub-characteristics. As an example maturity, availability, fault tolerance, and recoverability are sub-characteristics of the reliability category. Quality-in-use is composed of five characteristics, namely effectiveness, efficiency, satisfaction, freedom from risk, and context coverage.

Quality-in-use characterizes the impact of the software on stakeholders during a real usage. A platform owner looks for acceptable perceived experiences of use (efficiency), acceptable perceived results of use (effectiveness), acceptable perceived consequences of use (freedom from risks) and the customer’s satisfaction in a specific context of use (Herrera, Moraga, Caballero, & Calero, 2010). The information enhances the platform owner’s intuitions about the impact of the quality on users. Instead of separate measurements of quality-in-use attributes, an alternative approach is to translate all quality impact measures into a single measure that reflects perceived Quality of Experience (QoE), that the next section will discuss.

1.2.3 Quality of Experience

Quality of Experience (QoE) is a terminology borrowed from the telecommunications domain, used as a measure to determine how well the end users perceive to be satisfied with a particular feature, product, service or platform. “Quality of Experience (QoE) is the degree of delight or annoyance of the user of an application or service. It results from the fulfillment of his or her expectations with respect to the utility and / or enjoyment of the application or service in the light of the user’s personality and current state.” (Le Callet et al., 2012).

QoE is measured subjectively and objectively. QoE is measured subjectively where a user rats the perception of use based on emotion, experience and expectations. Subjective assessment of QoE is based on quantitative users’ ratings on a set of scales of momentary or remembered quality of features (Raake & Egger, 2014). For the subjective measure, Mean Opinion Score (MOS) is a known metric used by end-users to rate the service acceptability and quality perception. MOS is scaled ordinal usually in the range of 5 to 1 (Excellent, good, fair, bad, poor). However, the subjective assessment of QoE has challenges regarding the reliability of user ratings and involvement of a lot of users especially in a crowdsourcing scenario (Hoßfeld et al., 2014). To mitigate the challenges, objective measurements of QoE are used alternatively.

QoE can be predicted and measured objectively (Brooks & Hestnes, 2010) through objective measures relevant to Quality of Services (QoS) in the telecommunication domain such as end-to-end network quality, network coverage, suitability of service content and ease of service setup. QoE has been modeled in different application domains such as speech communication (Côté & Berger, 2014), audio transmission (Feiten, Garcia, Svensson, & Raake, 2014), video streaming (Garcia et al., 2014), web browsing (Strohmeier, Egger, Raake, Hoßfeld, & Schatz, 2014), mobile human-computer interaction (Schleicher, Westermann, & Reichmuth, 2014), and gaming (Beyer & Möller, 2014). The target application domain as the context of use defines metrics to model QoE.

To build an effective QoE control mechanism, objective and subjective measures are combined for a correlation analysis. A study shows a generic relationship between user-perceived quality (QoE) and network-caused quality (QoS) (Fiedler, Hossfeld, & Tran-Gia, 2010). This relationship can be used to estimate QoE for a certain value of the network quality and to identify the required quality level for achieving a specific level of QoE. We believe that this relationship is still valid where software quality replaces network quality and correlates with QoE. The current study will use the concept of this relationship.

QoE is influenced by several factors that can specify the reason for users’ perception in a particular experience. Human, system and context are categories of the influential and correlated factors that affect on QoE (Reiter et al., 2014). Human factors mainly address emotional attitudes, needs, motivations and expectations of human users. Context characterizes the user environment including physical, social, economical aspects. System factors determine technical quality of an application, a service or a device such as the performance of a mobile device, usability, and functionality of software products and availability of network communications. These factors will be important for interpretations of QoE results later on during assessments.

QoE and User Experience (UX) are two concepts with the centralization of experience of a human user. Wechsung & De Moor (Wechsung & De Moor, 2014) discussed that QoE and UX have many similarities but have more differences. The former is addressed in the telecommunication field and the latter in the Human Computer Interaction (HCI) field. QoE is mainly technology-centered where a large part of research around investigates Quality of Service. But, UX is human centered emphasizing on users emotions that is not driven by technology (Roto, Law, Vermeeren, & Hoonhout, 2011). The main focus of the QoE is on an evaluation of quality perception to inform optimization of technical parameters. But UX gathers inputs for designing products focusing on interactions for a pleasure experience. Despite the differences, the UX literature can be useful to support QoE-based research.

1.3 Research Objectives

A platform owner needs to gain insight into usage, health and sustainability of the platform to understand the risk factors that threaten the platform. Not only the external quality of platform services but also quality-in-use of the services in a software product are parts of risk factors. For the platform owner who needs to monitor the success of the platform, using a Quality-Impact assessment is a suitable approach. This approach assists the platform owner to assess the quality of platform together with its impact on Quality of Experience. The combination of analytics about software and human users together with qualitative user feedback contribute in designing the approach.

The overall aim of the research is to identify how the platform owner can use software quality analytics together with quality impact on users to gain insight into the success of a Future Internet Platform. To achieve the overall aim, the licentiate thesis defines the following objectives:

OBJ1: To understand software-human analytics that the platform owner uses for managing the platform.

OBJ2: To propose a holistic approach to the assessment of software products and services running on a Future Internet platform using the Quality-Impact approach.

> OBJ2.1: To describe the use of the Quality-Impact approach for elicitation of the appropriate level of software quality.

OBJ2.2: To describe a composition of analytics for holistic assessment of software products and services running on a Future Internet platform.

OBJ3: To validate the proposed approach in real-world practices.

OBJ3.1: To understand the effect of a feedback tool on the perceived quality of software products and services that the feedback is collected for.

The study investigates the objectives in three levels of acquiring knowledge, proposing a solution and validating the solution in real practices. Development of objectives relies on achievements of previous objectives, which indicates details of objectives had been defined along the progress of the study.

OBJ1 seeks for knowledge about software-human analytics and finding out the most relevant analytics to the objectives that different platform owners have determined. A literature study should be conducted to find out the research gaps in analytics and the state of practices. In the study, KPIs relevant to software ecosystem are explored. KPIs are qualified analytics that are aligned with the objectives of the platform owners. The research boundary of a software ecosystem assists understanding possible relevant analytics that are based on or enabled by software.

OBJ2 aims at an assessment of software products and services running on a Future Internet platform by using the generic relationship between quality of software and users' perception of the quality alongside the composition measurements in the lifecycle of software products and services. The choice of these groups of analytics comes from the OBJ1. OBJ2.1 aims at a Quality-Impact approach to elicit the relationships between quality and its impact on users. The approach uses principal knowledge outlined in the earlier work about generic relationships between Quality of Service and Quality of Experience and proposes a quality assessment approach. OBJ2.2 presents key ideas for a composition of measurements in the assessment of Future Internet products and services based on the use of analytics. The proposed approach models how to measure software quality analytics and predict user-perceived Quality of Experience in a Future Internet platform.

OBJ3 aims at real-world validations of proposed approaches in OBJ2. OBJ3.1 aims at understanding the side effects of feedback tools on QoE of the software products and services, which the feedback is requested for. This objective contributes to understanding characteristics of a feedback tool that may impact the perceived quality of the software.

1.4 Research Questions

The following research questions (RQ) have been formulated in the licentiate thesis to address the research objectives in section 1.3.

Table 1-1: The thesis's research questions and research outcomes

Research Questions[*]	Research Outcomes
RQ1: What analytics does a platform owner use to manage the success of a software ecosystem?	Classification maps of KPIs and platform owners' objectives in a software ecosystem.
RQ2: How can software products and services be assessed using a Quality-Impact approach in the Future Internet?	A holistic assessment approach using Quality-Impact relationship in the Future Internet.
RQ2.1. How can the relations between software quality and Quality of Experience be elicited?	Description of a Quality-Impact approach.
RQ2.2. How can analytics be composed for the assessment of software products and services in a Future Internet Platform?	Description of software-human analytics to be measured based on a composition model during the lifecycle of making products and services in the Future Internet.
RQ3: Does a feedback tool affect perceived quality of a software product and service?	Understand the effect of a disturbing feedback tool on QoE of software products and service. Disturbing characteristics of feedback tool are also parts of the outcomes.

* The labels of the research questions (RQ) are independent of the labels used in the studied papers, where each paper follows its own numbering schema.

To answer RQ1, Chapter 2 gives an overview of the literature that address a use of KPI in a software ecosystem. The relevant study identifies the purposes of using KPI in a software ecosystem and overviews the relevant KPIs to achieve the objectives. The result indicates the commonly used KPIs and objectives for managing a software ecosystem. To answer RQ1, the study answers the following questions:

- What kinds of ecosystems were studied?
- What types of research were performed?
- What objectives were KPI used for?
- What ecosystem entities and attributes did the KPI correspond to?

The answer to RQ2 proposes a Quality-Impact assessment approach for software products and services in the Future Internet based on findings in RQ2.1 and RQ2.2. The answer to RQ2.1 proposes a Quality-Impact approach to predict the quality level properly using quality impact analytics, and RQ2.2 proposes a composition of analytics to assess the quality impact using software quality analytics in a Future Internet platform.

RQ3 as an early step in the validation of the proposed solution evaluates the feedback tool used for data collection. RQ3 aims to study whether the triggered requests for feedback negatively affect user perception of the software quality. It also investigates the characteristics of the feedback tool may disturb users. To answer RQ3, Chapter 5 investigates the following research questions:

- Which characteristics of feedback requests did disturb users?
- How did disturbing feedback requests affect QoE of a software product?
- Did users provide feedback about feedback requests?

Figure 1-1 gives an overview of the included studies in the licentiate thesis. The figure shows research questions mapped to the outcome of each question in different phases of the study. Corresponding chapters to the research questions are also identified in the figure.

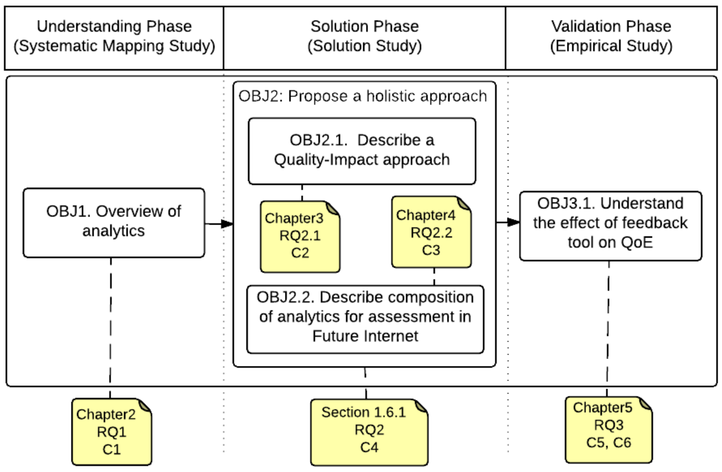

Understanding Phase (Systematic Mapping Study)
Solution Phase (Solution Study)
Validation Phase (Empirical Study)
OBJ2: Propose a holistic approach
OBJ2.1. Describe a Quality-Impact approach
OBJ1. Overview of analytics
Chapter3 RQ2.1 C2
Chapter4 RQ2.2 C3
OBJ3.1. Understand the effect of feedback tool on QoE
OBJ2.2. Describe composition of analytics for assessment in Future Internet
Chapter2 RQ1 C1
Section 1.6.1 RQ2 C4
Chapter5 RQ3 C5, C6

Figure 1-1: Overview of research studies

1.5 Research Methodology

This section presents the description of methodologies used in the licentiate thesis. *Systematic mapping study, solution proposal* and *empirical research* were the methods used in the licentiate thesis.

1.5.1 Systematic Mapping Study

To address RQ1, the study conducted a systematic mapping in Chapter 5. The systematic mapping approach gave an overview of KPIs used in a software ecosystem by classification of relevant articles and map the frequencies of publications over corresponding categories to build classification schemas and to see the current state of research (Petersen, Feldt, Mujtaba, & Mattsson, 2008). The systematic literature review was an alternative method for systematic mapping study. However, it differs in goals and depth. The aim of the study was not finding out the best practices based on empirical evidence. It was enough to have a broad overview rather than the time-consuming process of going through details in depth. The reasons motivated us to choose systematic mapping as the research methodology.

The research was conducted in the following four steps according to the guideline introduced in (Petersen et al., 2008): A database search, screening of papers, building classification schemas, and the systematic mapping of each paper. In the database search step, we defined the search string including keywords relevant to software ecosystem and KPI. The search strings were searched in software engineering and computer science research databases including Scopus, Inspec, and Compendex, which also support IEEEXplore and ACM Digital Library. In the screening step, we screened the identified papers to exclude studies that do not relate to the use of KPI for any ecosystem-related purpose. In the classification step, we employed keywording (Petersen et al., 2008) as a technique to build the classification scheme in a bottom-up manner. Extracted keywords were grouped under higher categories to make them more informative and to reduce the number of similar categories. In the last step, when the classification was in place, we calculated the frequencies of publication for each category and used x-y scatter plots with bubbles in category intersections to visualize the generated map.

1.5.2 Solution Proposal

The studies in Chapter 3 and Chapter 4 are solution proposals (Wieringa, Maiden, Mead, & Rolland, 2006). The studies propose a novel solution in the form of a technical approach for using Quality of Experience for assessment of software products and services. As recommended in solution proposal papers (Wieringa et al., 2006), the studies used examples or provided arguments as a proof-of-concept.

As required by a solution proposal, in both studies (Chapter 3 and Chapter 4) we explained why such novel approach was needed, specified the principles and steps of the method, and described how to apply the method.

1.5.3 Empirical Study

To address RQ3, the study conducted a mixed qualitative and quantitative research method as presented in Chapter 5**.** The empirical study used the QoE-Probe described in (Fotrousi, 2015) to collect user feedback for a requirement modeling software called Flexisketch (Golaszewski, 2013).

Participants: The participants were 35 software engineering students at the graduate level, and familiar with the concepts of requirement modeling. The study was performed as a part of the students' assignment in the Requirement Engineering course.

*Study procedure**:*** The procedure for each participant followed two parts:

1-*Software Usage*: Participants used Flexisketch integrated with the QoE probe as the feedback tool. In the QoE probe, the probability of automatic firing of the questionnaire was set to 10%. We requested the participants to model the requirements defined through a video in the Flexisketch and meanwhile provide answers to the fired questionnaire.

2-*Post Questionnaire*: At the end of the usage, we asked the participants to fill in the questionnaire about their feedback for the modeling tool as well as the feedback tool.

Data collection method: The feedback tool randomly requested participants to provide feedback automatically while working with the software product. The feedback tool collected ratings of the participants' experience with the feature they just used as well as their rationale for their choice. Also, the study collected participants' perceptions about feedback requests and about the experiences with the software product after the completion of the experience through a post-questionnaire. The collected feedback was analyzed individually to answer the research questions.

Data analysis method: The study used qualitative content analysis, pattern matching as well as quantitative descriptive analysis.

The study applied both *inductive* and *deductive* content analysis approaches (Elo & Kyngäs, 2008; Thomas, 2006). The inductive approach was *conventional* with the idea of coding data freely to generate information, and the deductive was *directed content* analysis approach based on using initial coding categories extract from the hypothesis but might be extended (Hsieh & Shannon, 2005). The study adapted the pattern matching analytical technique (Yin, 2014), by comparing a predicted pattern with observed patterns concluded empirically. The statistical correlation analysis was also applied to measure the relationships between observed variables.

1.5.4 Validity Evaluation

Similar to other research, the study is subject to validity evaluation.

Construct validity identifies whether the study reflects the phenomenon that was searched for. The validity threat in Chapter 2 addresses whether the included papers in the study reflected the Software ECOsystem (SECO) KPI phenomenon as it was intended to be researched. The search string captured the wide variety of software-related ecosystems with several names given to key performance indicators. The common databases used for software and management-related literature research including Scopus, Inspec, and Compendex, were used to find papers. Also, the list of included papers was validated against two systematic studies on software ecosystems (Barbosa & Alves, 2011; Manikas & Hansen, 2013b), and we found that our review covered all relevant papers.

The validity threat in Chapter 5 was relevant to the participation of students, in the sense that whether the students' answers were from their own perceptions, or whether they were based on what their teacher expected. To mitigate this threat, the assignment became optional, was not graded, and was just used as a part of the learning process.

Reliability validity refers to the repeatability of the study for other researchers. The study in Chapter 2 applied a defined search string and followed a step-by-step procedure that can be easily replicated. The stated inclusion and exclusion criteria were systematically applied. Reliability of the classification was obtained by seeking consensus among multiple researchers.

The validity threat was also discussable in Chapter 5 about repeatability of the content analysis. To mitigate this threat, the first two authors of the paper peer reviewed the quotes. The first reviewer documented the design of content analysis process as a guideline with the significant degrees of freedom for coding. To increase the reliability of the coding, the first and second authors, as reviewers, followed steps independently to achieve the same set of categories. In the case of some conflicts, they negotiated for the final categories.

Also, to increase the reliability of the results over evaluation of findings in Chapter 5 the study used triangulation strategy through content analysis, pattern matching and statistical analysis to answer the core research question (the second research question).

Internal validity refers to the extent to which the results may have been biased, and the study design avoids confounding. The threat is small in the study of Chapter 2, since only the descriptive statistics, which count the frequency of categories, were used.

External validity concerns the ability to generalize from this study. Generalization is not an aim of a systematic mapping study, as only one state of research is analyzed. In particular, the study results about the use of SECO KPI, reflects the practices studied in SECO KPI research and not SECO KPI practice performed in general.

Chapter 5 addresses the subject of external validity. The inductive content analysis targets a specific group of students that experience just one design-modeling product. To make the research generalized, similar research for other groups of population with different software should be designed and conducted as future research.

1.6 Results

1.6.1 Summary of Results and Solution Proposal

An overview of literature about KPIs in a software ecosystem in Chapter 2 revealed that platform owners mainly aim at *improving business, interconnectedness* between individual actors and subsystems of the ecosystem, as well as at *improving quality*. To assess the objectives they mostly use KPIs relevant to *satisfaction*, *performance* and *freedom from risks* measures (which answers RQ1).

Improving the quality and interconnectedness can be directly measured using quality-related analytics whether the analytics are relevant to software quality (e.g. *performance*, *freedom from risk*) or relevant to human (e.g. *satisfaction*). This relation informs Quality-Impact approaches.

A platform owner can use a Quality-Impact approach to elicit specific relationships between software quality levels and their impacts for given quality attributes on stakeholders as shown in Chapter 3 (which answers RQ2.1). In Chapter 3, an example of this relationship was discussed for eliciting non-functional requirements where an understanding of such relationship can specify the right level of quality for deciding about acceptable impacts. This approach proposed to measure software analytics objectively from a software product or services and subjectively from a formulated questionnaire through a workshop.

A platform owner measures the composition of analytics for assessment of products and services. The approach is proposed based on three models of measurement, composition and lifecycle as discussed in Chapter 4 (which answers RQ2.2). The measurement model describes measuring QoS and usage analytics (i.e. software analytics) together with QoE analytics (i.e. human analytics). Time, error and MOS (Mean Opinion Score) measures are valuable measures since they can support most of quality attributes defined according to ISO 25010. The measurement is applied during the lifecycle of the product (e.g. lab testing, runtime) based on rules defined for the propagation of quality measures according to the composition model.

Figure 1-2 illustrates an overview of a Quality-Impact approach (which answers RQ2) as the result of aggregating the answers to RQ2.1 and RQ2.2. The aim of the solution is to propose a holistic assessment approach that measures and analyzes the quality of software products and services as well as the impact of the quality on the users' feelings in during the software lifecycle in a Future Internet platform. The result of the analysis identifies how software products and services satisfy users and describes the acceptance level of their quality. The qualitative user feedback given after usage interprets the analysis. The assessment is not performed just for the final software product used in the real world environment, but it can also be performed during the factory acceptance testing, software release, and site acceptance testing.

Figure 1-2 shows a timeline that marks events relevant to data collection and analysis. Starting a user experience of a software product and service initiates collecting usage logs continuously during the usage period. The collected data aims to measure usage-based, time-based and error-based quality analytics. Requests for user feedback are fired periodically to collect the experience of a user during the experience at the end of the usage. A post-questionnaire collects the overall user feedback reflecting the user experience. The three types of data contribute to performing quantitative QoE and QoS analyses, correlation analysis between QoE and QoS, data as well qualitative feedback analysis. The analyses may cover all types of descriptive, predictive and perspective analytics that will be discussed amongst future research.

The data collection and analysis are performed based on using the composition model that defines which components have been integrated into the software product. The composition model in data collection identifies the source of measurements to be collected and in analysis identified how the quality is propagated between the involved components and infrastructure.

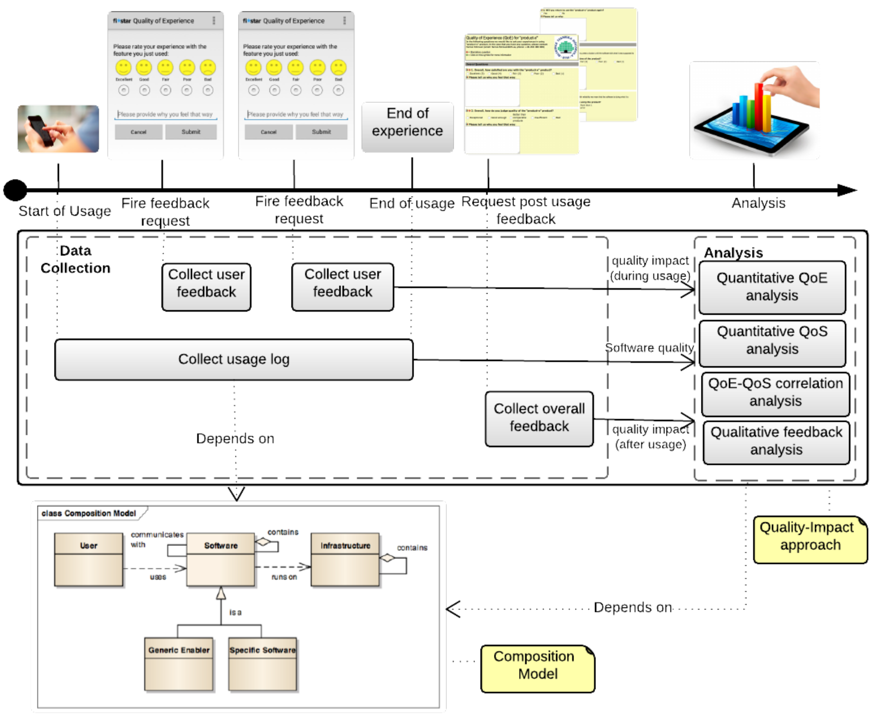


Figure 1-2. Overview of software products' assessment in a Future Internet Platform

To validate the solution, the thesis contributed to an implemented QoE-probe (Fotrousi, 2015) as a feedback tool developed for the Android operating system. The QoE probe is integrated with another mobile application to collect software analytics (QoS) as well as user ratings (QoE). During the integration phase, developers tag events relevant to start and end of features as well as important actions.

During the usage phase, the tags record the usage logs with data including application name, hashed user id, timestamp, event, feature name, action name to enable measuring usage and QoS analytics relevant to the product. Also, in the completion of a specific feature/scenario during the usage, a short QoE questionnaire is fired automatically to collect the overall user impression reflecting the user's experience. As recommended in (Menzies & Zimmermann, 2013), the feedback tool frequently asks users' opinions in a form of a short questionnaire:

Q: Please rate your experience with the feature you just used:

○*Excellent* ○*Good* ○*Fair* ○*Poor* ○*Bad*

Please provide why you feel that way: ____________________

In answering the questionnaire, QoE data are also logged. Together with the collected QoS data (usage log) it will be sent to the server for analysis.

As discussed in Chapter 5, the automatic firing of the feedback requests may disturb users. Feedback requests that interrupted a user task, too early in the process of learning the application, too frequently, or with apparently inappropriate content were perceived to be disturbing by the users.

However disturbing feedback requests did not necessarily affect users' perception of software product's quality (which answer RQ3). It means that if the feedback tool disturbs the users, it does not indicate that the QoE of the software products is always perceived bad. QoE of the software product was essentially justified with other influential factors such as quality of products and the devices that the product runs on.

1.6.2 Overview of Chapters

Chapter 2 gives an overview of the literature on the use of KPI (Key Performance Indicators) for software-based ecosystems. A systematic mapping methodology was followed and applied to 34 included studies published from 2004 onwards.

Two major kinds of ecosystems were researched: *software ecosystems* and *digital ecosystems*. Many application domains such as *software development, telecom, business management, logistics, transportation, healthcare* were addressed, but most of them with one or two papers only. The published research was mature with the journal, conference, and workshop papers. They were *conceptual proposal, solution proposal, validation, and evaluation papers* that covered *metrics, models,* and *methods* contributions.

The study showed that KPIs were used to achieve a variety of objectives. Platform owners aimed, at *improving business,* at *improving the interconnectedness between actors,* at *growing the ecosystem,* at *improving the quality of the ecosystem, product, or services performed within the ecosystem, and* at *enabling sustainability of the ecosystem*.

The included papers in this study described measurements applied to the whole ecosystem or a part of the ecosystem, which consists of *actor, artifact, service, relationship, transaction* and *network.* The measurement entities were identified in relation to the ecosystem objectives. To measure the entities, we classified the measurement attributes into categories of *size, diversity, satisfaction, performance, financial, freedom from risk, compatibility,* and *maintainability.*

Among the papers, the most common objectives were reported as *improving the interconnectedness* between *individual actors and subsystems of the ecosystem* as well as *improving quality* or *business* of the overall *ecosystem*. *Satisfaction, performance, and freedom from risks* measures were the most common categories of KPIs.

The presented taxonomy assists to identify relevant KPI considering domain and objectives, however due to the limitation of the study, the selection should be done carefully. The similarity of objectives is not the only factor in selecting a KPI.

The results of the mapping study indicate that more research is needed to obtain a better understanding of KPI, for software-based ecosystems. In particular, a deeper understanding of how the application domain affects an ecosystem's KPI is needed. Also, an important research opportunity is the identification, analysis, and evaluation of KPI. Such research could make the work with KPI more flexible because a greater variety of KPI would be known and available for the practitioner to use.

Chapter 3 describes an approach to quality requirements elicitation based on the inquiry of Quality-Impact relationships. The method, called Quality-Impact Inquiry, is based on the quality of a product and subjective feedback from the stakeholders about perceived quality impacts. The method guides a requirements engineer in the systematic inquiry of good-enough software quality from the viewpoint of the appropriate stakeholders of the software system. The solution proposal article describes the method in details and reports early experience from applying the method.

The Quality-Impact Inquiry method is performed in four processes: *preparation*, *measurement*, *analysis* and *decision-making*. During *preparation*, the required materials are prepared including the preparation and documentation of a prototype, the formulation of a questionnaire, the recruitment of stakeholders for participation in a workshop, and the scheduling of the workshop.

The *measurement* process aims at collecting quality measurements and user feedback. While stakeholders are using the software, quality attributes of the experience are measured. At the end of using the software, a questionnaire is administered to collect stakeholder opinions about the impacts of the perceived quality.

In the *analysis* process, the quality measurements are correlated with the stakeholder opinions about quality impact. The analysis uses a regression function to estimate the quality impact for a given quality value. The output of the analysis proposes a list of quality values for different quality impacts.

During the *decision-making* process, the requirement engineer decides about acceptable and desired levels of quality of the investigated quality attributes and updates for the relevant quality requirement in the SRS document, if needed. The step concludes with decision-making about whether to add inquiry iterations.

The Quality-Impact Inquiry method was applied in a real-world project in the healthcare domain. An example presented an application of the method for a Diabetes smartphone product. Diabetes patients use the product to take blood glucose measurements, to plan insulin injection, and to send the collected observation history to a diabetes specialist for consultation. The Quality-Impact relationship is evaluated for the features user authentication and observation sharing of diabetes information. The article reports how the method is applied to the requirements engineering endeavors, shares early experiences from applying the method, and gives recommendations reporting the practical use of the method.

Future research should aim at validating and evaluating the method in further, large-scale requirement engineering situations. Moreover, future research should aim at expanding the understanding of the generic relationships between given combinations of software quality attributes and their impacts as well as how quality attributes interact with each other. The resulting knowledge will translate into a SLA and help to allow and to reuse the knowledge of appropriate quality levels. It will also help accelerating and simplifying quality requirements inquiry in real-world projects, and enable research to check deeply held beliefs about how quality and impacts are interrelated.

Chapter 4 introduces quality assessment of heterogeneously sourced systems based on a composition of analytics. This approach combines QoS and QoE measurements to assess quality through the composition model of software during the software development lifecycle.

The paper introduces the concept of Generic Enablers (GE) as the generic components used to develop products for the Future Internet platform. It explains the characteristics of such heterogeneous sourced products in Future Internet together with the metaphor of a host that prepares a delicious meal for guests.

The paper compares the GE-based approach with buying the ingredients for a delicious home-prepared meal in a supermarket, where both the quality of the ingredients and their skillful preparation determine the quality of the prepared meal. Although the host can judge the quality of the meal, but the ultimate judgments of that quality is seen in the appraisals of the host’s guests and in the amount that people eat. By translation of this metaphor to the domain of the Future Internet, the paper discusses the importance of quality of GEs, the way they are composed as well as the corresponding ecosystem.

Then, the paper discusses three models: A measurement model, a composition model, and a lifecycle model. The approach addresses a measurement model that describes how analytics and empirical data is collected and used for the assertion of QoS and QoE. By the support of literature, the paper discusses the importance of time- and error-based measurements together with MOS measure. The paper also discusses how the measures support the quality characteristics of ISO-25010. Also, the approach proposes using the combination of the measurement and composition models to enable early measurement of quality. The approach proposes quality assessment in different stages of the lifecycle model. Measurements along the product lifecycle allow planning for step-wise improvement of the quality.

Future work includes validation of the approach. A particular focus will be given to the healthcare environment, where quality assurance is particularly important as it may decide on death or life.

Chapter 5 presents an empirical research that explores the relationship between feedback requests and QoE. In this study, the QoE-probe as the feedback tool was integrated with a mobile product. The feedback tool prompted users for the feedback about the product randomly. The tool collected users' perceptions of their experience during their interaction with the product. At the end of the experience, a post-questionnaire received users' feedback about the feedback tool and the software product.

The analysis of users' feedback about the feedback tool identified categories of causes that led to disturbance of users. Users perceived disturbance of Feedback requests that interrupted a user task, too early, too frequent, or with unsuitable contents. The findings contributed to parameterize the characteristics of feedback requests. The study modeled a feedback request with four-tuple variable referring to the experience space, the time frame within the space, number of feedback requests in the time frame and the content of feedback request. This model implies that user disturbance may be avoided by a suitable configuration of the variables.

The analysis of users' feedback about the software product showed the disturbances generated by the feedback tool have a negligible impact on the QoE of the software product. Triangulation of the study through three different analyses confirmed the finding: A pattern matching analysis showed that the disturbance caused by the feedback tool did not always create a bad experience of the software product. A correlation analysis confirmed that the QoE of the software product was not statistically correlated with the QoE of the feedback tool. The content analysis of users' feedback presented that QoE of the software product was essentially justified with other influential factors rather than via disturbing feedback requests. The quality of the software product and the experiencing context like device characteristics were the focal points of arguments to justify the users' rates.

The negligible impact of feedback requests on QoE implies that software product vendors may trust the data that their feedback tools collect, even if the feedback tool may disturb the users. The collected feedback can be informative about the software products or even the disturbing feedback tool.

The results of the study were limited to experiences of students with a modeling mobile product. The contextual factors might have an effect on the obtained results. In future, other studies with different types of products can complement the current study to increase the reliability of the results. Also, there is another open research in the future for empirical case studies where the parameters of feedback requests have been controlled in relation with QoE.

1.7 The Overall Contributions

This licentiate thesis provides the following six contributions:

C1: A better understanding of platform owners' objectives and relevant KPIs that are measured, analyzed and used for decision-making in a software ecosystem. From a researcher's point of view, the study captures state-of-knowledge and can be used to plan further research. From a practitioner's view, the generated map refers to studies that describe how to use KPI for managing of a SECO.

C2: Describing the Quality-Impact Inquiry method to elicit non-functional requirements based on relations between quality and its impact on users' perception. On one side, the method contributes to show how the correlation between quality and impact of the quality on users can be defined. On the other side, the Quality-Impact relationships can be used to design and dimension a software system appropriately and, in a next step, to develop service level agreements that allow for re-use of the obtained knowledge of good-enough quality.

C3: Describing the approach to measuring a combination of software-human analytics using a composition model during the lifecycle of making products and services in the Future Internet. The study contributes a QoS and QoE measurement-based approach to managing quality using a composition model during the lifecycle of products and services. The paper explains the approach with the metaphor of a host that prepares a delicious meal for guests. An exemplar is taken from the

FI-STAR project to describe how the approach is transferred into a real-world environment.

C4: Implementation of the QoE probe tool as an Android mobile product to collect product-human analytics as well as qualitative user feedback. The tool can be used in the development of feedback-based research projects as well as in the evaluation of software products based on users' feedback in practice.

C5: Understanding and modeling the characteristics of a feedback tool that may disturb users. The study parameterized the characteristics of feedback requests. It informs researchers with the factors that disrupt users' experiences in order to help them in finding feedback mechanisms to take control over users' disturbance. The study also helps practitioners in designing the feedback tool by adjusting the parameters.

C6: Understanding that feedback requests have negligible impacts on users' QoE of the software product as such. The study showed that the quality of software products have more impact on QoE rather than characteristics of the feedback tool. For practitioners, it implies the importance of their focus on the product's quality, although designing a proper feedback tool should not be neglected, since it contributes to collect informative feedback about the software product.

1.8 Conclusions and Future Work

The recent years have observed the development of heterogeneous composite software products and services that integrate common reusable functionalities from a Future Internet platform. Due to the propagation of the quality of such software, quality assessment and gaining insight into whether the quality fulfills the users' expectations are challenging. An analytics-driven approach that considers both software quality and the impact of the quality on users inform a platform owner to assess usage, health, and sustainability of the platform.

The study provided an overview of the literature on the use of analytics for software-based ecosystems. A produced classification map shows that platform owners mostly use *satisfaction*, *performance*, and *freedom from risk* categories of analytics. This finding was taken for proposing assessment solutions using analytics.

The study proposes a holistic approach for assessment of software products and services in a Future Internet platform using the Quality-Impact approach to gain insight into the success of software products and services. This approach measures and correlates the software quality analytics as well as the impacts on users' perception of the

Quality of Experience (QoE). The correlation analysis informs predicting QoE for a specific level of quality or find out the quality levels that may disturb the users. The approach uses the software composition model during data collection and analysis to model the Quality-Impact based on the propagation of the quality of the composite software.

To validate the Quality-Impact approach, a QoE probe was developed to collect software quality analytics automatically as well as users' QoE. The early validation of the Quality-Impact approach parameterized characteristics of feedback requests. Although disturbing feedback tools have negligible impacts on the perceived QoE of software products, designing the suitable feedback tool, however, contributes to collect effective feedback to enhance the products. The Quality-Impact approach was adopted as a part of the validation of the FI-PPP project. To validate the Quality-Impact solution, a QoE probe was developed in order to collect software quality analytics automatically as well as users' Quality of Experience. The early validation of the Quality-Impact approach parameterized characteristics of feedback requests. Although disturbing feedback tools have negligible impacts on the perceived QoE of software products, designing a suitable feedback tool contributes to collect effective feedback to enhance the products. The Quality-Impact approach was adopted as a part of the validation of FI-PPP project.

Future work aims at validation of the proposed Quality-Impact assessment approach through empirical research such as case studies in the FI-PPP project or other real practices. Future research will investigate how to correlate QoS and usage analytics with the QoE analytics and how the qualitative user feedback will be supportive of the interpretation of quality. The analysis will consider the composition model to investigate how quality propagates in the composite software. In this correlation analysis, all three types of descriptive, predictive and perspective analytics will be investigated.

Furthermore, the study aims at improving the design of the QoE probe to be supportive of the above study. A self-adaptive mechanism of receiving feedback, which applies lessons learned about suitable characteristics of a feedback tool, is planned in the study's roadmap. The future research also extends the capability of QoE probe to collect data in a distributed environment where elements of the composite software provide services through distributed nodes in a Future Internet platform.

CHAPTER 2 KPIs for Software Ecosystem: A Systematic Mapping Study

Abstract

To create value with a software ecosystem (SECO), a platform owner has to ensure that the SECO is healthy and sustainable. Key Performance Indicators (KPI) are used to assess whether and how well such objectives are met and what the platform owner can do to improve. This paper gives an overview of existing research on KPI-based SECO assessment using a systematic mapping of research publications. The study identified 34 relevant publications for which KPI research and KPI practice were extracted and mapped. It describes the strengths and gaps of the research published so far, and describes what KPI are measured, analyzed, and used for decision-making from the researcher's point of view. For the researcher, the maps thus capture state-of-knowledge and can be used to plan further research. For practitioners, the generated map points to studies that describe how to use KPI for managing of a SECO.

Keywords

Software ecosystem, digital ecosystem, performance indicator, KPI, success factor, systematic mapping

2.1 Introduction

A software ecosystem (SECO) is about "the interaction of a set of actors functioning as a unit and interacting with a shared market for software and services, together with the relationship among them" (Jansen, Finkelstein, & Brinkkemper, 2009). We include here any ecosystem that is based on or enabled by software, including pure software, software-intensive systems, mobile applications, cloud, telecommunications, and digital software ecosystems. The inclusion of telecommunications, for example, is important as many modern software services can only be realized with appropriate ICT infrastructure. Companies adopt a SECO strategy to expand their organizational boundaries, to share their platforms and resources with third parties, and to define new business models (Manikas & Hansen, 2013b; Weiblen, Giessmann, Bonakdar, & Eisert, 2012).

A SECO is frequently supported by a technological platform or market that enables the SECO actors in exchanging information, resources, and artefacts. Ownership of such a platform gives strategic advantages over the other SECO actors. It allows satisfying ever-increasing customer demands with limited own resources. It also allows improving one's own knowledge about the marketplace. Such knowledge is necessary for innovation, evolution of a product or service offering, and identification of revenue opportunities (Barbosa & Alves, 2011; IBosch, 2009).

SECO platform ownership also brings responsibilities. These include the definition of SECO performance objectives and management of the SECO to achieve these objectives. A SECO is expected to be healthy (Costanza & Mageau, 1999) and sustainable (Chapin et al., 1996). It is healthy when it is productive for surrounding actors, robust, and niche-creating (Iansiti & Richards, 2006). It is sustainable when it maintains its structure and functioning in a resilient manner (Costanza & Mageau, 1999). Health and sustainability are closely linked performance objectives (Rapport, Costanza, & McMichael, 1998) that are often found in complex systems (Costanza, 1992).

Managing a SECO involves definition of how actors, software, and business models play together to achieve the SECO objectives (Manikas & Hansen, 2013a) in business, technical, and social dimensional perspectives (Santos, Werner, Barbosa, & Alves, 2012). The platform owner uses performance indicators for benchmarking and monitoring the resulting ecosystem behavior. Key performance indicators (KPI) are those among the many possible indicators that are important, easily measurable quantitatively or with an approximation of qualitative phenomena (Parmenter, 2010). The KPI serve as early warnings about potentially missed SECO objectives (Westin, 1998) and to detect patterns that are useful for predicting health and sustainability of the SECO (Cokins, 2009). Any deviation from success baselines are recorded and acted upon to ensure that the main ecosystem's objectives are met.

The here presented study gives an overview of literature on KPI for software ecosystems. A systematic mapping methodology was followed to identify and classify publications based on the reported research and based on KPI use. The dimensions used for classifying research were the type of ecosystem that was studied and the type of result that was delivered by the research. The dimensions used for classifying KPI use were the researched KPI types, the SECO objectives these KPI were used for.

The knowledge gap for collecting evidences about KPI studies motivated to systematically evaluate distribution of studies and provide guidance for future improvement. For practitioners, the generated map describes how to use KPI in the management of a SECO. It enables the platform owner in understanding the indicators that are important to assess for given SECO objectives. For researchers, the generated map describes state of research and helps finding research gaps for understanding the definition and use of SECO KPI.

The remainder of the paper is structured as follows. Section 2 presents the research objectives and defines research questions, search strategy, study selection, and study quality assessment. Sections 3 and 4 present the results by giving an overview of SECO KPI research, respectively SECO KPI practice. Section 5 discusses the results. Section 6 summarizes and concludes.

2.2 Research Methodology

The goal of this study is to provide an overview of the research performed to investigate the use of KPI for managing software ecosystems. The systematic mapping approach (Petersen et al., 2008) allows to map the frequencies of publications over categories to see the current state of research. It also exposes patterns or trends of what kind of research is done, respectively has been ignored so far. Mapping the research results, in addition to the type of research, reveals researchers' current understanding of KPI-related practice.

2.2.1 Research Questions

To provide an overview on publications relevant to KPI use for SECO, two sets of research questions are defined in Table 2-1. With the first set of questions we mapped foci and gaps of research about SECO KPI. With the second set we mapped the state of practice that was reported by the research.

Table 2-1: Research questions

SECO KPI Research	**Rationale**
RQ1: What kinds of ecosystems were studied?	The answer to this question shows the intensity of SECO KPI research across application domains and types of ecosystems. Skewedness, e.g. due to a focus on just a few types of application domains and ecosystems, indicates gaps where additional research is needed.
RQ2: What types of research were performed?	The answer to this question shows the maturity of SECO KPI research. The more disproportioned conceptual solutions and empirical validation research are, the more there is a need for research that compensates.
Ecosystem KPI Practice	**Rationale**
RQ3: What objectives were KPI used for?	The answer to this question shows the purposes of SECO KPI. It allows understanding when a SECO is considered to be successful and when not. Correlation with the answer to RQ4 allows understanding how the satisfaction of these SECO objectives is measured.
RQ4: What ecosystem entities and attributes did the KPI correspond to?	The answer to this question gives an overview of relevant KPI that are used to assess achievement of SECO objectives. The KPI show how SECO objectives are operationalized and quantified. Skewedness, a focus on just one or a few KPI, may indicate the degree of universality the KPI have for SECO management.

2.2.2 Systematic Mapping Approach

To answer RQ1, RQ3, we followed the systematic mapping guidelines proposed by Petersen (Petersen et al., 2008). We (i) conducted database search with a search string that matched our research scope, (ii) performed screening to select the relevant papers, (iii) built a classification scheme based on keywording the papers' titles, abstracts, and keywords, and (iv) used this classification scheme to map the papers. To answer RQ2, we modified the mapping process by using the pre-existing classification schemes already used in (Petersen et al., 2008; Wieringa et al., 2006). For RQ4, we built the classification scheme by extracting keywords from the main body of the papers and aligning the emerging scheme with the relevant software industry standard. The research steps are explained below.

(i) Database search. The study defined the following search strategy.

Search String. To get an unbiased overview of KPI use in SECO, the search string was created with keywords that capture population only. The first aspect used to define the population was the ecosystems that can be found in a software context: software, digital, mobile, service, cloud, telecommunication, and ICT ecosystems. We also included papers that focused on software supply by adding software supply to the search string. The second aspect used to define the population was the application or use of KPI. We used the terms indicators, metrics, measurements, success factors, key characteristics, and quality attributes as synonyms for KPI. To avoid bias about RQ3, we did neither constrain for what purpose information was gathered and used. To build a broad overview of the research area and avoid bias, no keywords were defined in relation to intervention (e.g. monitoring), outcomes (e.g. improvements to a SECO), or study designs (e.g. case studies).

The search string was built by concatenating the two population aspects with the *AND* operator. The search string was formulated as follows: *software OR (software-intensive) OR digital OR mobile OR service OR cloud OR communic* OR telecom* OR ict) PRE/0 (ecosystem* OR "supply network*") AND (measur* OR kpi* OR metric* OR analytic* OR indicator* OR "success factor*" OR "quality attribute*" OR "key characteristic*".*

Search Strategy. The papers were identified using the important research databases in software engineering and computer science including Scopus, Inspec, and Compendex, which support IEEEXplore and ACM Digital Library as well. The search string was applied to title, author's keywords and abstract of these papers. The search did not restrict the date of the publication.

Validation. We validated the set of identified papers by checking it against the papers used in the SECO literature reviews performed by (Barbosa & Alves, 2011; Manikas & Hansen, 2013b). Each paper used by these studies that was relevant for our study had been found by following the above-outlined database search.

(ii) Screening of papers. The inputs for this step were the set of papers identified with step (i). The first and second authors screened these papers independently We screened these papers to exclude studies that do not relate to the use of KPI for any ecosystem-related purpose and to ensure broad-enough coverage of the population. We describe here a complete set of inclusion and exclusion criteria.

Inclusion. We included peer-reviewed journal, conference, or workshop papers that were accessible with full text. The included papers describe the use of KPI in an ecosystem context or the effects of such KPI on properties of the ecosystem. Due to the importance of networking infrastructure and digital information exchange for a well-functioning software ecosystem we included telecommunication and information technology papers in addition to pure SECO papers.

Exclusion. We excluded papers that focused on the use of KPI for managing a member of the ecosystem only. For example, papers about the use of indicators for managing a single company that participates in the ecosystem, or a product or process of that company, were excluded because of their too narrow focus. We excluded papers that focused on other ecosystems rather than a software ecosystem. For example papers focus on biology, environmental, climate, and chemical aspects were excluded. When the definition of software ecosystem did not fulfill in the papers, they were excluded. As an example, the paper that considered Bugzilla and email system as software ecosystems was excluded, since such systems do not address the shared market concept of a SECO definition. Papers that study qualitative indicators using qualitative approaches such as a structured interview were excluded. Also, we excluded papers that focused on ecosystem design in place of ecosystem management. For example, papers about the design of interoperability protocols or of products or services offered to an ecosystem were excluded. The papers that do not Finally, to avoid inclusion of papers that only speculated about KPI use or effects, we excluded papers that did not report any empirically-grounded proof-of-concept.

(iii) Building the classification scheme. To answer the research questions RQ1, RQ3, and RQ4 we employed keywording (Petersen et al., 2008) as a technique to build the classification scheme in a bottom-up manner. Extracted Keywords were grouped under higher categories to make categories more informative and to reduce number of similar categories. We built the ecosystem classification scheme by extracting the types and application domains of the studied ecosystems. We built the classification scheme for KPI practice by extracting KPI assessment objectives, entities and attributes used for measuring the KPI.

The keywords were extracted from the papers' titles, keywords, and abstracts. When the quality of an abstract was too poor, we used the main body of the paper to identify the keywords. Similarly, as most of the papers did not included sufficient information about entities and attributes measured with KPI inside the abstract, we used the main body of the papers for keyword identification. The keywords obtained from extraction were then combined and clustered to build the categories used for mapping the papers. The clustering of measurement attributes was aligned with the categories described in ISO/IEC FDIS 25010 as far as applicable.

To answer RQ2, we used a pre-defined classification scheme (Wieringa et al., 2006) that was used by earlier systematic mapping studies (Petersen et al., 2008). It classifies research types into validation research, evaluation research, solution proposals, philosophical papers, opinion papers, and experience papers.

(iv) Systematic mapping of the papers. When the classification scheme was in place, the selected papers were sorted into the classification scheme. The classifications were then calculated the frequencies of publications for each category.

To answer RQ1 and RQ2 we reported the frequencies of the selected papers for the categories in the dimensions of ecosystems types and application domains, respectively in the dimensions of research type and research contributes type. We used x-y scatterplots with bubbles in category intersections to visualize the kinds of ecosystems that were studied. The size of a bubble is depicted proportional to the number of papers that are in the pair of categories that correspond to the bubble coordinates. The visualized frequencies make it possible to see which categories have been emphasized in past research and which categories received little or no attention.

To answer RQ3, we first described the categories identified when building the classification scheme and how these categories were expressed in the selected papers. This description resulted in a dictionary for interpreting the scatterplots used for describing how SECO KPI are used in relation to these objectives. We again used x-y scatterplots for showing the frequency of pairs of categories. These pairs allowed us to describe the attributes measured for each type of ecosystem entity, the measurements used in relation to the SECO objectives, and how KPI are obtained for various kinds of entities found in a SECO.

2.2.3 Threats to Validity

This section analyzes the threats to validity for the taxonomies of construct, reliability, internal and external validity.

Construct validity reflects whether the papers included in the study reflect the SECO KPI phenomenon that was intended to be researched. The search string was constructed in an inclusive manner so that it captured the wide variety of software-related ecosystems and the many different names given to key performance indicators. The common databases, used for software and management-related literature research, were used to find papers. Only after this inclusive process, manual screening was performed to exclude papers not related to the research objectives. The list of included papers was then validated against two systematic studies on software ecosystem (Barbosa & Alves, 2011; Manikas & Hansen, 2013b) and found that the review covers all relevant papers.

Reliability validity refers to the repeatability of the study for other researchers. The study applied a defined search string, used deterministic databases, and followed a step-by-step procedure that can be easily replicated. The stated inclusion and exclusion criteria were systematically applied. Reliability of the classification was achieved by seeking consensus among multiple researchers.

Internal validity treats refers to problems in the analysis of the data. These threats are small, since only descriptive statistics were used.

External validity concerns the ability to generalize from this study. Generalization is not an aim of a systematic mapping study as only one state of research is analyzed and the relevant body of research completely covered. In particular, the study results about the use of SECO KPI, reflects the practices studied in SECO KPI research and not SECO KPI practice performed in general.

2.3 Results: Ecosystem KPI Research

The database search resulted in a total of 262 papers, including 46 duplicates. After screening and exclusion, 34 papers remained and were included in the study. These selected papers were published from 2004 onwards. This section gives an overview of the research described in the selected papers. Appendix A lists the selected papers.

2.3.1 Kinds of Ecosystems

To answer RQ1, Figure 2-1 gives an overview over the ecosystems that our study found KPI research for. The number embedded in a bubble indicates how many papers were devoted to a given combination of ecosystem type and application domain (multiple classifications possible). Empty cells indicate that no corresponding study was found. The number on the category label indicates the total number of papers in that category.

Most of the papers used the term software ecosystem to characterize the studied ecosystems. Special kinds of ecosystems were cloud, service, mobile apps, and open source software ecosystems. Less frequent were digital ecosystems with 44% of the papers. They refer to the use of IT to enable collaboration and knowledge exchange (Boley & Chang, 2007).

The papers addressed a variety of application domains. Most common were telecommunications, business management and software development. None of the remaining application domains was addressed by more than one or two papers. Thus research is rather scattered, and the specifics of the various application domains only little understood.

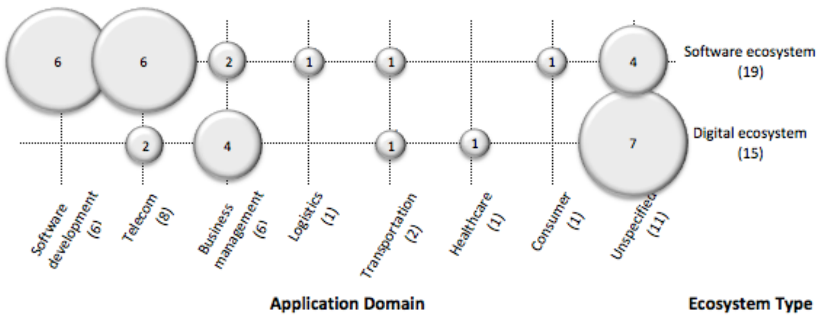


Figure 2-1: Kinds of ecosystems that were studied with KPI research. The label "software ecosystem" refers to those that are not considered a digital ecosystem (see main text).

2.3.2 Types of Research

To answer RQ2, Figure 2-2 presents a map of the kind of research performed on KPI in software-related ecosystems. Papers with multiple research types and contributions were classified for each combination of research type and contribution they presented.

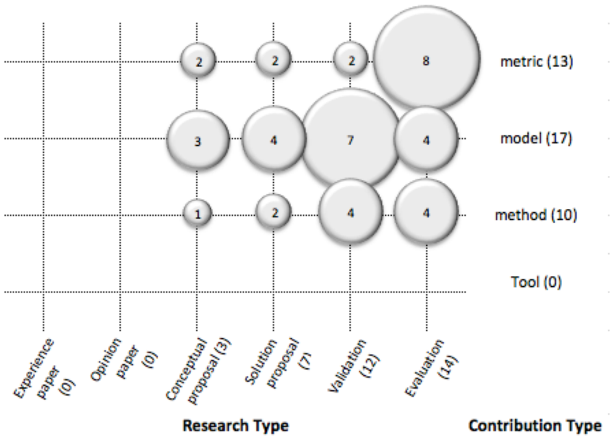


Figure 2-2: Map of research on SECO KPI and type of contributions.

Experience report papers describe experiences in working with SECO KPI and usually describe unsolved problems. *Opinion papers* discuss opinions of the papers' authors. *Conceptual proposal* papers sketch new conceptual perspectives related to SECO KPI. This category renamed *philosophical papers* category (described in iii of section 2.2) to fit the SECO KPI study. *Solution proposal* papers propose new techniques or improve existing techniques using a small example or a good argumentation. *Validation* papers investigate novel solutions that had not been implemented in practice (e.g. experiment, lab working). *Evaluation* papers report on empirical or formal studies performed to implement a solution or evaluate the implementation.

Metric papers describe KPI for SECO. *Model* papers describe relationships between KPI. *Method* papers describe approaches for working with SECO KPI. Finally, *tool* papers describe support for work with SECO KPI.

Most research was found in the categories of validation and evaluation. Research contributed with metrics, models, or methods. For example, R17 proposes a model that explains how health can be measured with relevant indicators (conceptual proposal, model) and validates that model with a questionnaire (validation, model). R14 proposes a method for assessing services based on Quality of Service indicators (solution, method). R19 evaluates factors that affect successful selling in e-markets (metric, evaluation). No paper was an experience report or an opinion paper. No paper contributed with any tool.

2.4 Results: Researched KPI Practice

The papers included in this study describe the use of KPI by a platform owner for achieving objectives with the ecosystem that was enabled by the ecosystem platform. This section gives an overview of these objectives and the KPI that were used.

2.4.1 Ecosystem Objectives Supported by KPI

KPI were used to enable or achieve a variety of objectives. Platform owners aimed, at improving business, at improving the interconnectedness between actors, at growing the ecosystem, at improving quality of ecosystem, product, or services performed within the ecosystem, and at enabling sustainability of the ecosystem (answer RQ3):

Business improvement. Research has been performed on how to improve business at the ecosystem level. The studied business improvements concerned the perspectives of ecosystem activity and of commercial success. Ecosystem activity related to the level of activity of participating actors, encouragement to participate in the ecosystem, and the transaction volume. Commercial success related to sales success, innovativeness and competitiveness of the participating actors, and the cost of the network that enables the ecosystem. The activity and commercial perspectives were mixed in the papers, thus could not be separated in the analysis of the literature.

Interconnectedness improvement. Research has been performed on how to improve interaction in an ecosystem, for example to reduce cost, improve predictability of services that are provided in the ecosystem, and manage trust. Interaction improvement was studied between individual actors and between whole networks contained in the ecosystem. The research differed in terms of lifecycle stage of an interaction and covered supplier availability, discovery, ranking and selection, the resulting connectivity, interaction evaluation, and the impact of the interaction on the actors that participated in it. Interaction improvement was not always an end in itself, but was considered essential for generating business activity and sustainability of the ecosystem.

Growth and stability. Research has been performed on how to manage growth and stability of the ecosystem. Growth and stability were seen as two factors that need to be managed jointly. During growth flexibility and controllability need to be maintained. During stability, a continuous co-revolution must happen. Growth and stability again are not ends in themselves, but thus contribute to sustainability and survival of the ecosystem.

Quality improvement. Research has been performed on how to manage quality of ecosystems. In particular, performance, usability, security, data reliability, extendibility, transparence, trustworthiness, and quality-in-use were investigated. Quality management was sometimes presented as an ends in itself, for example by allowing comparison among multiple ecosystems, enabling diagnosis, improving decision-making, and achieving long-term usage of services. At the same time, however, quality management was considered to be a means to encourage adoption and growth, improve business performance, and achieve sustainability.

Enable sustainability. Research has been performed on how to sustain an ecosystem. Two angles were taken: self-organization and resource consumption. Self-organization was approached through continuous rejuvenation of the ecosystem. Resource consumption was studied in relation of electrical energy. Throughout all papers found in this category, sustainability was considered to be desirable ends for software ecosystems.

2.4.2 KPI: Measured Entities

The included papers describe measurements applied to the ecosystem as a whole or the parts the ecosystem consists of: actor, artifact, service, relationship, transaction and network.

Actors. Actors were measured and characterized as follows. They were human or artificial. Examples of human or legal actors were sellers and developers that provide products to buyers or groups of organizations and firms. Examples of artificial actors were nodes in a telecommunication network. An actor engages in transactions in an ecosystem and builds relationships to other actors or artifacts. The transactions the seller engages in generate profit and revenue for the cost the seller is willing to take. Effective actors have knowledge about other actors or the network and has good interestingness and reputation for other actors. Actors are also considered to be sources and sinks of data and have differing ranges for data transmission. Performance of individuals and groups in terms of fulfilled tasks and decisions as well as performance of firms and organizations in terms of profits are measured.

Artifacts. Artifacts such as software, codes, plugins, books, music, or data were measured and characterized as follows. Artifacts had a location in the ecosystem. They evolve, may have reputation and popularity, and exposed their consumers to vulnerability.

Services. Services were measured and characterized as follows. Services consume energy and other resources. Services have quality attributes such as quality of service, security, compliance, and reputation. Metadata and service level agreements are used to specify the services. The services are not fixed but evolve: services emerge, change, and get extinct. A special service was provided by the platform that laid the fundament for the ecosystem. It was characterized in terms of attributes like stability, documentation, portability, and openness.

Relationship. Relationships were measured and characterized as follows. Actors enter relationships with other actors, artifacts, or services. A relationship connects two or more such entities. Examples of relationships were business connections and telecommunication communication links. A relationship may be transparent and express a trust value of the connected entities. A relationship is the basis for transactions, thus is used for advertising and building alliances. The transaction, however, is constrained by cost and quality of the relationship.

Transactions. Transactions were measured and characterized as follows. Examples of transactions are sales of services to customers, server requests, and commits of code files made by developers. They are initiated with an offer that is measured in terms of attributes like price and quantity. Transactions also have a price and quantity. Other attributes include time to negotiate the transaction, time to complete, energy consumption, transmission rate, and buyer satisfaction.

Network. Networks were considered as sets of entities and relationships that were part of a whole ecosystem. Examples were local or application-specific networks. Networks were characterized as follows. Networks were vulnerable to security threats such as data availability, integrity, authentication, and authorization. Networks differed in the node density, degree of collaboration, provisioning cost, and hit rate for artifacts.

Ecosystem. Full ecosystems were characterized as follows. They have quality attributes like size, performance, security and energy consumption that can also characterize networks contained in an ecosystem. In addition, ecosystems exhibited lifelines, diversity, stability, transparency, healthiness, and sustainability.

This section and next section collaboratively provide answer for RQ4. The map in the left part of Figure 2-3 shows the entities that were studied in relation to the ecosystem objectives. Most research studied the measurement of the overall ecosystem to enable quality or business improvement. For example, R17 describes how performance of the ecosystem affected user satisfaction, and R13 shows how analytics applied to the ecosystem can be used to improve business. Considerable research was also devoted to improving the interconnectedness of the ecosystem, where attributes of the products and services played an important role and also to the role of platform measurements to grow the ecosystem and improve quality. For example, R6 described how to use a service similarity measurement was used to improve ecosystem connectivity. R2 described how growth, diversity, and entropy measurements of a SOA platform were used to increase growth. R4 described how communication quality measurements were used to improve the quality of a telecommunication ecosystem.

The map also shows areas where no research was published. For example no research studied the role of network measurements for objectives other than sustainability and quality improvement.

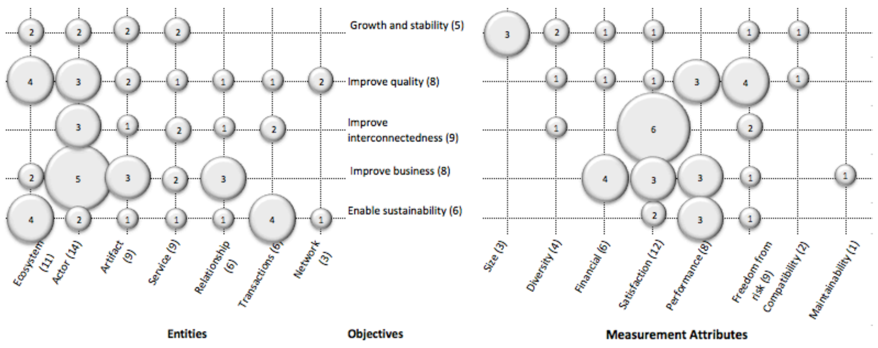


Figure 2-3: Map of measured entities and measurement attributes in relation to ecosystem objectives.

2.4.3 KPI: Measurement Attributes

To make the state and evolution of the ecosystem and of its elements visible, a broad variety of attributes were measured.

The following attributes categories emerged when clustering the attributes described in the included papers. Figure 2-4 shows how classes of quality attributes were merged toward new categories. The *size* category includes attributes to measure size and growth. *Diversity* includes attributes to measure heterogeneity and openness for such heterogeneity. *Financial* includes attributes to measure economic aspects such as investment, cost, and price. *Satisfaction* includes attributes to measure satisfaction and the related concepts of suitability, interestingness, learnability, usability, accessibility, acceptability, trust, and reputation. *Performance* includes attributes to measure performance, including resource utilization, efficiency, accuracy, and effectiveness. *Freedom from risk* includes attributes to measure the ability to avoid or mitigate risks and includes the related concerns of security, reliability, maturity, availability, and other related guarantees. *Compatibility* includes attributes to measure the degree to which an entity can perform well in a given context, interoperate or exchange information with other entities, and be ported from one context to another one. *Maintainability* includes attributes to measure flexibility, respectively the ability to be changed.

The right part of Figure 2-3 gives an overview of the attributes referred to by KPI. Most research studied measurements of satisfaction, typically to improve business or interconnectedness. An example of such research is R13 that describes the use of seller reputation to improve business. To support quality improvement, all measurement attributes that relate to quality were included in at least one research paper, except for maintainability and size. Similarly, size measurements did not play any role other than for growth and stability.

- *Diversity*
    - Heterogeneity
    - Openness
- *Satisfaction*
    - Satisfaction
    - Suitability
    - Interestingness
    - Learnability
    - Usability
    - Accessibility
    - Acceptability
    - Trust
    - Reputation.
- *Performance*
    - Performance
    - Resource utilization
    - Efficiency
    - Accuracy
    - Effectiveness
- *Financial*
    - Investment
    - Cost
    - Price
- *Size*
    - Size
    - Growth
- *Freedom from risk*
    - Risk mitigation
    - Security
    - Reliability
    - Maturity
    - Availability
    - Guarantees.
- *Compatibility*
    - Interoperability
    - Exchangeability
- *Maintainability*
    - Flexibility
    - Changeability

Figure 2-4: Merging classifications of measurement attributes

The left part of Figure 2-5 shows how the ecosystem elements were measured. Satisfaction was a common attribute that was measured for any entity except for rules. This shows that a same attribute can be measured or analyzed for different ecosystem entities. Also it is revealed that similar measurement attributes might be collaborating to measure different ecosystem elements. As an example CCCI (correlation, commitment, clarity and importance) measurable attributes were used to measure trust as well as reliability.

The overall ecosystem and actors were the most comprehensively measured or analyzed entities, with a special focus on satisfaction, freedom from risks and performance. Some examples of such satisfaction measurements are provided by R13 that measured usage and acceptability of an ecosystem. The service followed with the next largest variety of measurements. R2, for example, measured entropy and diversity to characterize platform complexity. Only narrow sets of measurement attributes were applied to the business partner, interactions, and business.

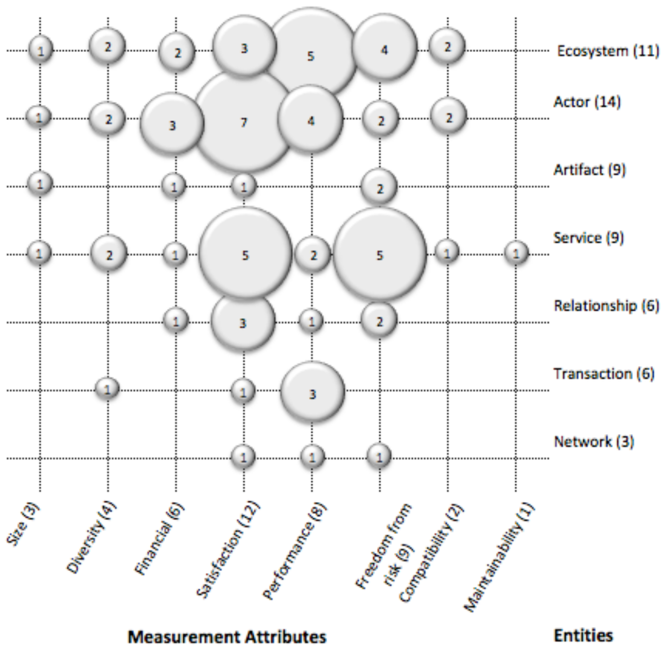


Figure 2-5: Map of measurement attributes in relation to the measured entities.

2.5 Discussion

The study provides a classification of KPI relevant papers in understanding researches, relationship with the practice, and assessment of research outcomes. This classification contributes to taxonomy, which can help for closer examination of the ecosystem or platform owner objectives, making them more recognizable in designing KPI. New KPI can be extracted for an ecosystem using this taxonomy, and existing KPIs can be extended or restructured applying the generic structure of the taxonomy.

The literature map indicates that KPI for software-based ecosystems is a thin area with work at all maturity levels. Journal, conference, and workshop papers exist. However, the number of publications is not sufficient, and many application domains for ecosystems addressed with just one or two papers. Although formulation of KPI might be domain dependent and similarity of objectives is not the only factor to select a KPI, however due to insufficient study it is difficult to state whether characteristics of a domain, for example regulation of healthcare, affects the KPI of the ecosystem that targets that domain.

The included research on ecosystem KPI mostly addresses ecosystem measurements or measurements of satisfaction, performance and freedom from risks. Measurements other than satisfaction that are applied on elements contained in the ecosystem are comparatively little researched. A broader understanding of KPI would increase a platform owner's flexibility in measuring, analyzing, and using KPI for decision-support. The understanding of a greater variety of KPI would also contribute to increased transparency of status, evolution, and other aspects of the ecosystem.

2.6 Conclusions

The here presented study gives an overview of literature on the use of KPI for software-based ecosystems. A systematic mapping methodology was followed and applied to 34 included studies published from 2004 onwards.

To respond to RQ1 and RQ2, research was broad but thin. Two major kinds of ecosystems were researched: software ecosystems and digital ecosystems. Many application domains were addressed, but most of them with one or two papers only. The published research was mature with journal, conference, and workshop papers that covered metrics, models, and methods. In response to RQ3 and RQ4, KPI research was skewed. Most research studied ecosystem KPI for improving the interconnectedness between individual actors and subsystems of the ecosystem. Overall, most KPI were about satisfaction, performance and freedom from risks measures.

The results of the mapping study indicate that more research is needed to better understanding of KPI for software-based ecosystems. In particular, a deeper understanding of how the application domain affects an ecosystem's KPI is needed. Also, an important research opportunity is the identification, analysis, and evaluation of KPI. Such research could make the work with KPI more flexible, because a greater variety of KPI would be known and available for the practitioner to use.

CHAPTER 3 Quality Requirements Elicitation based on Inquiry of Quality-Impact Relationship

Abstract

Quality requirements, an important class of non-functional requirements, are inherently difficult to elicit. Particularly challenging is the definition of good-enough quality. The problem cannot be avoided though, because hitting the right quality level is critical. Too little quality leads to churn for the software product. Excessive quality generates unnecessary cost and drains the resources of the operating platform. To address this problem, we propose to elicit the specific relationships between software quality levels and their impacts for given quality attributes and stakeholders. An understanding of each such relationship can then be used to specify the right level of quality by deciding about acceptable impacts. The Quality-Impact relationships can be used to design and dimension a software system appropriately and, in a second step, to develop service level agreements that allow re-use of the obtained knowledge of good-enough quality. This paper describes an approach to elicit such quality–impact relationships and to use them for specifying quality requirements. The approach has been applied with user representatives in requirements workshops and used for determining Quality of Service (QoS) requirements based the involved users' Quality of Experience (QoE). The paper describes the approach in detail and reports early experiences from applying the approach.

Keywords

Requirement elicitation, quality attributes, non-functional requirements, quality of experience (QoE), quality of service (QoS)

3.1 Introduction

Quality requirements are an important class of non-functional requirements (Glinz, 2007). They concern software system attributes such as functional suitability, performance, reliability, usability, security, and portability that are important for achieving stakeholder goals (Boegh, 2008). The satisfaction of these quality attributes determines whether the software system meets the goals of its stakeholders or whether the system has a negative impact for these stakeholders (Chung, Nixon, Yu, & Mylopoulos, 2000; Haigh, 2010).

Meeting the right level of quality is important to balance benefits and cost (Regnell, Berntsson Svensson, & Olsson, 2008). The quality of a software system needs to be at least as good as to make the software useful and competitive, but should not be excessive to avoid cost and unnecessary use of resources. Insufficient quality leads to disappointment and consequent churn when stakeholders decide to abandon the software solution and adopt alternatives instead (Kilkki, 2008). Excessive quality may lead to an unnecessarily expensive design of the software system (Bass, Clements, & Kazman, 2012), to unnecessary consumption of resources needed for operating the system (Jung, Hiltunen, Joshi, Schlichting, & Pu, 2010), and to trade-offs where other quality attributes suffer (Braz, Seffa, & M'Raihi, 2007).

To address the problem of finding the level of good-enough quality, the relationship between software quality and the impacts of such quality for the stakeholders of the software system needs to be understood. As demonstrated for the Quality of Service (QoS) of a telecommunication network and the Quality of Experience (QoE) of the network users, a quality–impact relationship can be developed empirically by setting quality levels of a given quality attribute and measuring the reaction of the stakeholders that were exposed to these quality levels (Fiedler et al., 2010).

This paper describes how to use quality–impact analysis for eliciting requirements about good-enough quality of a software system. The proposed method guides the elicitation of the quality–impact relationships and explains how to use the gained insights to specify quality requirements. The method delivers empirical evidence for a specific software system that is more reliable than generic expert opinion. The evidence pertains to the features that were investigated and the stakeholders that were participating in the requirements inquiry, thus is adequate and relevant for decision-making about that software system's quality requirements.

The paper describes the proposed quality–impact elicitation method in depth. It gives details about the key ideas of the method and explains how to tailor the method depending on the investigated quality characteristics, the stakeholder goals impacted by these quality characteristics, and the instruments that the investigator is able to apply. The paper provides an example of how the method is applied in practice by reporting about its use in a real-world software development project.

The paper is structured as follows. Section II reviews existing work and motivates quality requirements elicitation based on quality–impact relationship inquiry. Section III describes the method in-depth. Section IV describes how the method is applied and reports the lessons-learned from such method application. Section V compares the method and the obtained results with related work. Section VI concludes.

3.2 Related Work

According to ISO/IEC FDIS 25010, the quality of a system is the degree to which the system satisfies the needs of its stakeholders. The determination of whether a system exhibits the desired quality characteristics is not straightforward, however. In contrast to functional requirements, many quality requirements do not have a sharp boundary between satisfaction and non-satisfaction. Instead, they are gradually satisfied (Glinz, 2005), thus called soft requirements (Irvine & Levin, 2000).

The softness characteristic of implies that the right level of desired requirements quality needs to be identified during requirements engineering (Regnell et al., 2008). Each such quality level has its own specific costs and benefits. High quality levels are considered more costly than low quality levels because more expensive designs or approaches to provision of the software service need to be chosen to implement the requirement. In a similar vein, increase of the quality level implies increase of the benefits generated by the requirement. A product that is considered useless because of too low quality becomes useful or even competitive with increased quality. Too much quality, however, is considered excessive thus not adding any value for stakeholders despite quality improvement. The trade-off between cost and value impacts is a basis to determine the desired quality level and specify the requirement in a quantified manner (Gilb, 2005; Jacobs, 1999).

Goal models have been proposed to elicit quality requirements (Antón & Potts, 1998; Chung et al., 2000). Such models allow identification of needs for improving, increasing, or keeping the level of the quality characteristics of a software. To support systematic identification of goals and qualities within a given domain, ontologies have been developed and used to support requirements elicitation (Souag, Salinesi,

& Wattiau, 2012; Wang et al., 2010). The means-ends relationships that are an inherent part of a goal model make the impact of such a quality requirements explicit (Cysneiros & Sampaio do Prado Leite, 2004; Herrmann & Paech, 2008). The goals that are enabled by such a decision are used as a rationale that motivates the quality requirement.

Unfortunately, goal models are of limited help eliciting appropriate levels of quality. Goal models help identifying the quality characteristics that are perceived relevant by stakeholders, and the means-ends relationships connect these qualities to the impact that is desired by the stakeholders. However, they do not guide a requirements engineer in how much of a desired quality is good enough. One of the key limitation is that the goal models do not relate a given quality level to a given level of impact beyond the coarse-grained levels of a requirement being denied, weakly denied, undecided, weakly satisfied, and satisficed . In addition, the application of goal models does not deliver the information needed to quantify a quality requirement, thus make its satisfaction measurable with attributes such as scale and meter (Jacobs, 1999).

Several supporting elicitation methods have been proposed for requirements elicitation (Pohl & Rupp, 2011). These include the use of questionnaires, interviews, workshops, creativity methods, storyboards, use cases, role-plays, and prototyping. Review of prototypes has been particularly effective in identifying usability concerns and refining user interaction design to reach user acceptance (Rettig, 1994). The construction of such prototypes allows a development team to capture assumptions about desired software characteristics and to validate these assumptions, for example by reviewing them as implementation proposals with concerned stakeholders (Fricker & Glinz, 2010; Fricker, Gorschek, Byman, & Schmidle, 2010).

The supporting elicitation methods provide limited support for the determination of good-enough quality levels because of their generality. Any question can be asked in a questionnaire or interview, any topic explored in a workshop, and a multitude of design decisions be captured with storyboards, use cases, role-play, and prototypes. Guidelines that have been proposed to identify quality requirements (Hassenzahl, Wessler, & Hamborg, 2001; Kusters, van Solingen, & Trienekens, 1999) target the discovery of quality, but do not help in determining measurable levels of quality. The requirements engineer is thus left with his intuition or experience for asking the right questions (Doerr, Kerkow, Koenig, Olsson, & Suzuki, 2005). The use of experience, however, is risky as the levels for good-enough quality may change between different software products and product-usage contexts.

To enable requirements engineers to determine appropriate levels of good-enough system quality, we were studying the field of telecommunication. In particular we were looking for approaches that allow the requirements engineer and the system stakeholders understand the meaning of a given level of quality, for example in terms of how the quality level affects the degree of stakeholder satisfaction. In the field of telecommunication, substantial work has been performed for understanding how to measure degrees system quality and how a given degree of system quality affects user attitude (Fiedler et al., 2010).

For a telecommunication system, Quality of Service (QoS) requirements are stated that concern system performance, availability, and capacity (Wang et al., 2010). Often these requirements are agreed in a service level agreements (SLA) between the system customers and the supplier (Kittlaus & Clough, 2009). User satisfaction is expressed as Quality of Experience (QoE) and refers to the "degree of delight or annoyance of the user of an application or service" (Le Callet et al., 2012). It has been shown that a system's Quality of Service affects the user's Quality of experience (Fiedler et al., 2010). Too little user delight and too much user annoyance leads to churn, thus users that try to look for alternatives and try to avoid using the system under consideration.

The knowledge of how QoS is related to QoE has not been translated into requirements engineering methodology yet. In particular, it is unclear how to exploit the relationship between QoS levels with QoE levels in the inquiry of software systems requirements. Also needed is an explanation of how to apply the specifics of the QoS-QoE relationship on the determination of good-enough quality for any system quality attribute and for any important stakeholder need that is impacted by the possible quality levels.

3.3 Quality-Impact Inquiry

This paper proposes a method that we call Quality-Impact Inquiry to address the so far unsatisfactorily solved problem of determining adequate levels of quality. As required from a solution proposal, we have explained why a novel method was needed, specify the principles and steps of the method, and describe how to apply it (Wieringa et al., 2006). To demonstrate that the method is sound, we go a step further than required from a solution paper and report about a preliminary validation that we performed with a real-world software development project. The paper describes the method in sufficient depth to enable replication in practice and further validation research.

The Quality-Impact Inquiry method is based on the principles outlined in our earlier work about the generic relationships between Quality of

Service and Quality of Experience (Fiedler et al., 2010). These principles have been translated into a software requirements engineering context by integrating it into an inquiry-based requirements analysis process (Potts, Takahashi, & Antón, 1994) and combined with prototyping, questionnaires, and workshops as supporting methods for collection of quality measurements and stakeholder opinions. During the workshop, stakeholders are exposed to requirements engineer-defined quality that has been implemented in the prototype and questioned about their perceived quality impact. The correlation between quality measurement and stakeholder opinion is analyzed and used as decision-support to determine and then specify good-enough requirements quality.

The Quality-Impact Inquiry method adapts the inquiry cycle of requirement analysis (Potts et al., 1994) as follows: the documentation phase is adapted to implement a prototype using a set of accepted requirements described the desired system and collects quality attributes during stakeholder actions. The three elements of requirement discussion phase including questions, answers and reasons are supported by the questionnaire elicitation. Finally the results from the former phases contribute to either freeze or change requirements in the evolution phase.

Figure 3.1 gives an overview of the Quality-Impact Inquiry process. The remainder of this section describes the generic Quality-Impact Inquiry method and how the method may be tailored. The ensuing section describes how the method has been applied in real-world projects and reports about early lessons-learned.

3.3.1 Inquiry Process

Figure 3.1 gives an overview over the process that characterizes the Quality-Impact Inquiry method. The process contains four steps: preparation, measurement, analysis and decision-making. It is applied iteratively until enough evidence has been collected to decide about what good-enough quality should be for a quality attribute under investigation.

1) Preparation: During the first step, Preparation, the materials needed for allowing stakeholders to experience the quality characteristics under investigation are prepared. The work includes the preparation and documentation of a prototype, the formulation of a questionnaire, the recruitment of stakeholders for participation in a workshop, and the scheduling of the workshop.

In the proposed method, quality impact is measured subjectively through a questionnaire. The quality impact is also affected by a real value of quality that is measured objectively (Brooks & Hestnes, 2010) and automatically using a prototype. Therefore a list of valid quality requirements are identified from SRS document that is relevant to one feature or a group of features (f) and presented as pairs of quality attribute and value:

$$Q = \{ (q_{att}, q_{val}) \mid f \} \quad (1)$$

As an example in SRS, a non-functional requirement can be stated as "response time should be less that 2 s". "Response time" is the attribute and 2 s is the value.

The software might be in a preliminary release (i.e. pre-alpha, alpha and beta testing), a candidate release close to a final product/service, or even a released product ready for an evolution. Preparation of artifacts including a prototype from a software feature(s) and a questionnaire about their quality is the pre-requisite to run the method. The stakeholders experience the software and then answer the questionnaire. Data that are collected from the software use and the questionnaire are analyzed to evolve quality requirements in the software specification document (SRS) if needed.

Based on the quality attributes, the prototype is tailored for the feature(s) f to support measurement of Q. The questionnaire will be tailored using Q to collect quality impacts of feature(s) f relevant to user list U:

$$U = \{u\} \quad (2)$$

Then, scenarios for data collection, and software guidelines to be followed by users are prepared in this step. Translating the

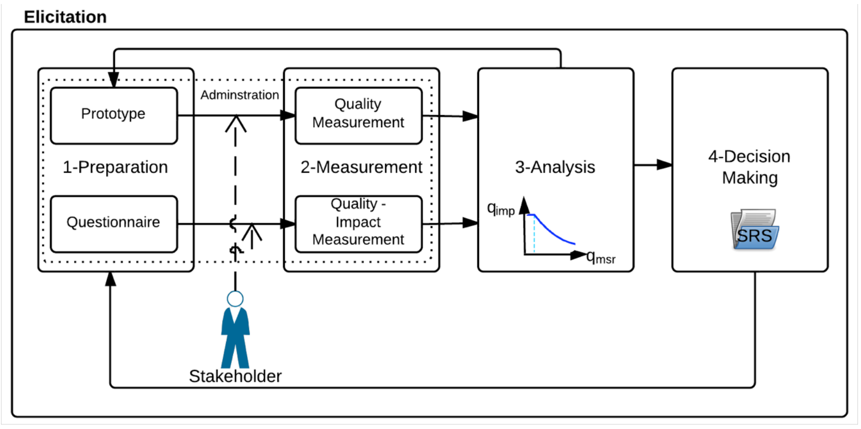


Figure 3.1: Quality-Impact Inquiry Method

questionnaire to the user's mother tongue is another action that might be required.

2) Measurement: During the second step, Measurement, a workshop is performed with the aim of collecting quality measurements and user feedback. During the workshop the stakeholders experience predetermined qualities by utilizing the prototype according to a pre-defined script. During the use of the prototype measurements are taken about the quality that the stakeholders experienced. After the use of the prototype, the prepared questionnaire is administered to collect stakeholder opinions about the impacts of the perceived quality.

While the users are using the application through clients such as a smartphone or a PC, quality values q_{msr} (i.e. q_{msr} is a q_{val} relevant to q_{att} for feature(s) *f*) are quantified by function m, automatically using analytical tools, server log generators or piece of codes embedded in the software.

$$q_{msr} = OP(\, m(\, q_{att} \mid u, f\,)\,) \mid OP \in \{\, MIN \text{ or } MAX \,\} \quad (3)$$

The function captures the worst value of measured quality attributes in different actions of a user for the given feature(s) f, depending on whether the quality has a success or failure measure characteristic (Fiedler & Hoßfeld, 2010). For a success measure such as availability, the higher value of the quality attribute shows better quality but for a failure measure such as response time, a higher value of the quality attribute shows worst quality. Therefore minimum or maximum value of each case would be the candidate value for measured quality.

Another source of measurement is the questionnaire designed to translate the quality impacts q_{imp} (i.e. q_{imp} is a q_{val} relevant to q_{att} for feature *f*) into scored values provided by users. In the questionnaire, users are typically asked to provide ratings,

$$q_{imp} = s(\, q_{att} \mid u\,, f\,) \quad (4)$$

and rationales in forms of comments that explain their ratings:

$$comm = c(\, q_{att} \mid u\,, f\,) \quad (5)$$

Furthermore, the questionnaire asks users to rate "quality in use" attributes such as satisfaction as a sub list of quality attributes:

$$Q_{inUse} \subset Q \quad (6)$$

The quality impact is translated into a discrete value that is scaled using scores such as Mean Opinion Score (MOS) (ITU-T, 2003).

3) Analysis: During the third step, Analysis, the quality measurements are correlated with the stakeholder opinions about quality impact. This step involves application of statistical analyses based on data that has been collected during the measurement step in the ongoing and previous Quality-Impact Inquiry iterations. The analysis can also be enhanced through a-priori knowledge of the generic nature of the studied Quality-Impact relationships.

The relation between the measured quality (q_{msr}) and quality impact (q_{imp}) will be identified through a regression analysis, similar to correlation analysis between QoE and QoS (H.-J. Kim et al., 2008; Minhas & Fiedler, 2013). The regression function is calculated for a feature f and quality attribute qatt:

$$\hat{q}_{imp}(q_{msr}) = r(q_{msr} \mid f, q_{att}) \quad (7)$$

Different regression functions for the relationship including linear, logarithmic, exponential and power have potential to be candidate, however the analysis compares the regression function and matches the best one.

Then, an estimation of quality value for a given quality impact is calculated by the inverse function of the regression model:

$$\hat{q}_{msr}(q_{imp}) = r^{-1}(q_{imp} \mid f, q_{att}) \quad (8)$$

The output of the analysis proposes a list of quality values for different quality impacts including maximum quality impact.

If the Quality-Impact analysis does not provide enough data for a mature analysis, some changes on the prototype are applied to change the quality values artificially. The looped arrow from analysis box to prototyping box in Figure 3.1 provides possibilities to achieve enough data for investigating impact changes and perform more reliable analysis than the analysis of less data points.

4) Decision-Making: During the fourth step, Decision-Making, the analysis results are used to decide about acceptable and desired levels of quality of the investigated quality attributes. The decisions are recorded in the software requirements specification. The step concludes with decision-making about whether to add inquiry iterations and how the parameters of these ensuing inquiries should be adapted for best improving the knowledge about good-enough quality.

The decision-making process selects suitable quality value from the evidences and decides whether to evolve the value for the relevant quality requirement in the SRS document.

This process identifies maximum applicable quality impact considering technical feasibility, product strategies, and limitation of resources to achieve the relevant quality value, and then applies the decision making function.

Decision-making is a function of parameters including estimated quality value for maximum impact ($\hat{q}_{msr}$) of a quality attribute, the value of relevant non-functional requirement (q_{SRS}), the list of rationale for the quality attribute rating (*comm*) beside all quality-in-use ratings (Q_{inUse}), to interpret whether the current quality fulfills the users acceptance.

$$q_{new} = \{\ g(\ \hat{q}_{msr}, q_{SRS}, comm\ , Q_{inUse} \mid f, q_{att})\ \}\textit{The thesis also contribute to a holistic approach} \quad (9)$$

This function defines a new value for the quality attribute. The decision-making will be performed for all quality attributes in *Q*.

3.3.2 Method Tailoring

There are a wide variety of variation points to adapt the generic Quality-Impact Inquiry process. The variations are needed to be flexible enough to adapt the process to specific requirements engineering constellations. Table 3-1 gives an overview.

Table 3-1. An overview of variations

Variation Point	Variants
Software Features	Stakeholders may be exposed to different features. Quality requirements may be specific to features or the impact of quality levels be perceived differently depending on the feature.
Quality Attributes	Stakeholders may be exposed to different quality attributes. Each feature or application may have its own set of prioritized quality attributes.
Quality Levels	For the selected quality attributes, different quality levels may be investigated. The selection of the quality level should be based on information need and be guided by statistical analysis methodology.
Stakeholder Sampling	Different individuals may be invited for participation in the inquiry workshops. The selected stakeholders should be as representative as possible.
Impact Attributes	Stakeholders may be questioned about different quality impacts. Each application or feature may aim at achieving its own specific impacts.
Measurements	Different measurements may be selected to record quality levels and stakeholder impacts.
Prototyping Approaches	The simulation of different quality characteristics may require different approaches of building the quality-simulating prototype.

Variation Point	Variants
Impact Function	Different impact functions may be chosen the represent the relationship between a given quality attribute and its impact. We were using linear and exponential functions so far.

3.4 Real-World Example of Method Application

3.4.1 Example Application

To demonstrate how to implement the method in practical situations, we present here the results and lessons-learned of an early validation that we have done in a real-world project. We applied the method for a Diabetes Smartphone Application that will be used by diabetes patients to take blood glucose measurements, to plan insulin injection, and to send the collected observation history to a diabetes specialist for consultation. We evaluated the Quality-Impact relationships for the features user authentication and observation sharing of diabetes information.

As an input to the Quality-Impact inquiry we had used a prototype that was instrumented with software for monitoring the timing of user interactions. The inquiry was performed in a laboratory and with a smart phone from the application developers with pre-loaded data. The requirements engineer, the product manager, and selected end-users participated in the inquiry workshop. The inquiry was performed with one end-user at the time.

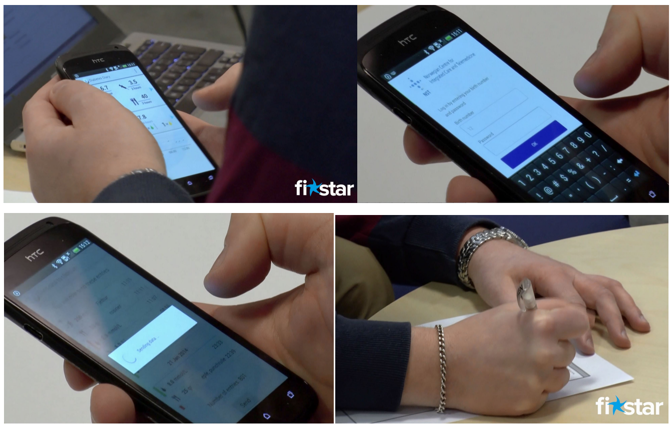


Figure 3-2: User interaction scenario with instrumented application and subsequent answering of the quality of experience questionnaire

During the inquiry, the end-user was introduced to the tasks he to be performed with the application, was given a short, tailored user manual, and then used the selected features first according to instructions and then without help. He opened the application, selected the data he wanted to share with his clinician, authenticated himself, and submitted the data. Then the authentication service requested username and password. When authenticated, the data was sent to the application server in the hospital. After the guided and unguided experiences were concluded, the end-user filled out the quality of experience questionnaire. Figure 3-2 gives an impression of the setup.

The Quality-Impact inquiry processes was implemented for the Diabetes Smartphone Application as follows:

1) Preparation: The requirements engineer extracted relevant quality requirements from the software requirement specification document. Based on these extracts he instrumented the software with a time-stamp logger.

The requirements engineer created a short guideline to assist the end-user in using the application. It described the features to be evaluated and how the features should be used.

Based on the extracted quality requirements, the requirements engineer created a quality of experience questionnaire with generic questions about the experience, about the features and product, and about the perceived quality. For the Diabetes Smartphone Application, the quality questions were about performance, reliability, and availability. Figure 3-3 shows the questionnaire.

2) Measurement: The following steps describe the inquiry workshop that was performed once for each user separately.

In the beginning of the inquiry the requirements engineer welcomed the participants, defined the goals of the inquiry, and shared the agenda of the meeting.

The product manager explained the feature to be used and gave prepared guideline to the end-user.

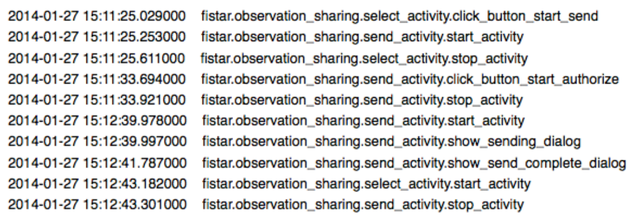
The end-user used the application according to the instructions. He did so twice to allow us collecting data about the learning and knowledgeable use of the feature. The application generated logs automatically and captured information from the user interaction (see Figure 3-4 for an example). In all timestamp, the time from the internal clock on smartphone was used. Log entries were created when end-user requests are received and when application screen/data have been displayed. The response time extracted from the example is the duration between two time stamps taken from the starting to the ending of an activity.

The Experience
1. Please tell us the name you would give to the feature:
The Features and Product
2 Overall, how satisfied are you with the features you just have experienced? □ Excellent (5) □ Good (4) □ Fair (3) □ Poor (2) □ Bad (1) Please tell us why you feel that way:
3. Overall, how good is the feature according to your opinion? □ Exceptional □ Better than comparable products and features □ Good-enough □ Insufficient Please tell us why you feel that way:
4. Will you return to use the product again? □ Yes □ No Please tell us why you feel that way:
The Quality
5. The next question is about response time. With response time we mean the time when you press a button until the software does what it is supposed to do. How do you rate the response time of the feature? □ Excellent (5) □ Good (4) □ Fair (3) □ Poor (2) □ Bad (1) Please tell us why you feel that way:

Figure 3-3: Questionnaire. The last question can be replicated and adapted to any feature the requirement engineer is interested of.

After application usage, the requirements engineer provided instructions for answering the quality of experience questionnaire. The user answered the questionnaire accordingly. The answers that were collected with quantitative scales provided data for calculating the Quality-Impact relationship. The qualitative rationale that the users gave for these values assisted us in interpreting the quantitative values.

At the end of the session, the requirements engineer debriefed the participants and thanked them for the participation.



```
2014-01-27 15:11:25.029000    fistar.observation_sharing.select_activity.click_button_start_send
2014-01-27 15:11:25.253000    fistar.observation_sharing.send_activity.start_activity
2014-01-27 15:11:25.611000    fistar.observation_sharing.select_activity.stop_activity
2014-01-27 15:11:33.694000    fistar.observation_sharing.send_activity.click_button_start_authorize
2014-01-27 15:11:33.921000    fistar.observation_sharing.send_activity.stop_activity
2014-01-27 15:12:39.978000    fistar.observation_sharing.send_activity.start_activity
2014-01-27 15:12:39.997000    fistar.observation_sharing.send_activity.show_sending_dialog
2014-01-27 15:12:41.787000    fistar.observation_sharing.send_activity.show_send_complete_dialog
2014-01-27 15:12:43.182000    fistar.observation_sharing.select_activity.start_activity
2014-01-27 15:12:43.301000    fistar.observation_sharing.send_activity.stop_activity
```

Figure 3-4: Extract from the log file with timestamps and activities

3) Analysis: The filled-in questionnaires and time-stamp logs from all end-users interactions were the inputs for the analysis process. The end-users were satisfied with the quality as they reflected in the questionnaire Therefore the analysis of this example did not identify any deviation to update quality attributes. However the similar study was conducted in our lab where users perception of response time in downloading a webpage containing an image were collected (Shaikh, Fiedler, & Collange, 2010). The analysis of the data distributions concluded a close match for a regression formula on relations between MOS and response time excluding null opinion scores:

$$\hat{q}_{imp}(q_{msr}) = 4.836 \exp(-0.15\, q_{msr}) \quad (10)$$

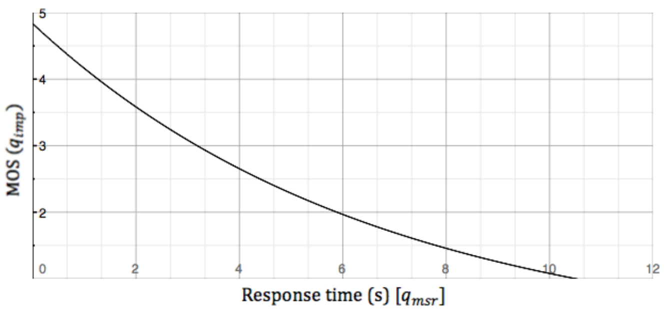


Figure 3-5: Quality impact (MOS) as a function of quality value (response time(s))

Figure 3-5 plots this regression function that shows quality (q_{imp}) as a function of quality value (q_{msr}) (Shaikh et al., 2010). The response time collected from different experiments as well as collected relevant quality impacts will plot the Figure 3-6. Taking the reverse of this function estimates quality value ($\hat{q}_{msr}$) as a function of quality impact (q_{imp}):

$$\hat{q}_{msr} (q_{imp}) = -6.67 \ln(q_{imp} / 4.836)s \quad (11)$$

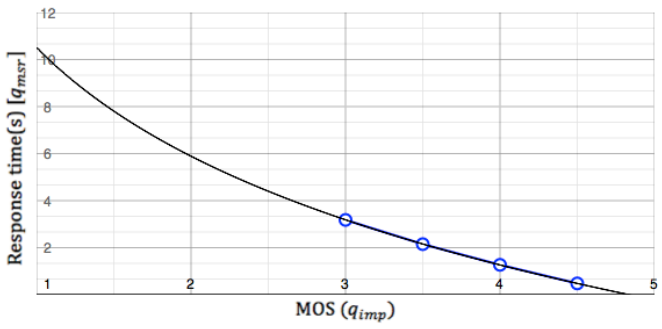


Figure 3-6: Quality value (Response time (s)) as a function of quality impact (MOS)

As Figure 3-6 plots the inverse regression function, shorter response identifies the better perception of quality and user's score. Table 3-2 estimates quality values for qimp in the range of between 3 and 4.5. This value identifies the best threshold value for a quality attribute such as response time that is sufficient for the user expectations.

Table 3-2. Estimated quality values for given quality impacts

Quality impact (q_{imp}) MOS	Estimated quality value ($\hat{q}_{msr}$) for Response time
4.5	0.48 s
4	1.27 s
3.5	2.15 s
3	3.18 s

4) Decision making: Decision making process involves choosing a threshold value for a quality attribute based on inputs from analysis including an estimated quality value for response time, user experiences and rationales, the list of quality-in-use as well as the value of response time defined in the SRS document.

Selecting the good-enough quality level requires trade-offs between the reaching enough user acceptance level instead of maximum level in return for gaining technical feasibility by limited resources such as cost, time and effort. Identifying maximum applicable user perception (quality impact) in each analysis is the result of such trade-offs. If quality impact 4 is recognized enough, then the estimated quality value of 1.27s will be involved in decision making process to update SRS with a good-enough quality value. Typically, the critical value for quality impact is assumed to be 3. In telecommunication area, accepted quality impact in video streaming is considered as 3.5, although the quality impact of 4 is a good choice (Khan, Sun, Jammeh, & Ifeachor, 2010).

3.5 Lesson learned

As shown in the example, the inquiry workshop allowed us to collect the data necessary for analyzing the Quality-Impact relationship for response time and quality of experience. The workshop lasted about 10 minutes per user. Data aggregation and analysis was concluded within a few hours. Thus the method was relatively efficient. Scalability can be achieved by working with multiple users in parallel, for example as part of a training workshop.

From the users that participated in the inquiry workshops we received positive feedback about the experience and about most of the questions we asked. However, one of the users was puzzled about perceived reliability and availability. He stated that he expected the application to work and to be available in the laboratory situation he was invited to. This shows that usage context affects the relevance of quality attributes. Some quality attributes are relevant in some contexts only. We plan to account for this feedback by extending the Quality-Impact inquiry to prolonged pilot uses of the application in the real-world contexts of the users.

On little usages of the software product could not give the full impression to users. An issue relevant to a quality attribute such as availability might not be risen in a short period of use, this is what reflected by the stakeholder in the example stated in section IV. To reach more accurate data, a prolonged usage should be planned.

Not only quality attributes are identified in the proposed Quality-Impact inquiry method, there might be some proposals for updating functional requirements extractable from the users' comments given in the questionnaire. As an example, if the end-user could not find how to submit the blood glucose data, this could be reflected in the users' perception rating as well as provided rationale.

Training before and during the workshop provides knowledge and skills to mitigate the threats of biasing the user perception that occurred due to misuse of the feature. Distractions during the workshop should be removed to boost concentration of users in expressing their real unbiased perception.

3.6 Discussion

The Quality-Impact Inquiry method is a generic approach to collecting data about quality levels and how these quality levels impact stakeholder satisfaction. It builds on our earlier work that shows that a relationship between quality levels and quality impact can be established. The Quality-Impact Inquiry method extends such earlier work by describing a 4-step process that allows the requirements engineer to inquire how different levels of quality impact the satisfaction of stakeholder needs. The 4-step process is independent of the specific type of quality and independent of the specific kind of stakeholder need. Instead the method can be tailored to any pair of quality and impact measurement that are of interest for the system under consideration. A condition for such tailoring is that a relationship between quality level measurements and impact measurements can be established.

The identified level of quality impact transforms the knowledge into a judgment of good-enough quality. Good enough quality can be decided considering cost and benefit views while exposing barriers and breakpoints (Regnell et al., 2008). Product strategy decisions, competitors and learning processes are other factors that assist requirement engineer to adjust the level of quality.

The Quality-Impact Inquiry method complements existing quality requirements elicitation methods. Pairs of system quality and impact variables that should be investigated as part of requirements inquiry can be identified with goal-based inquiry methods (Chung et al., 2000; Potts et al., 1994). Means-ends relationships of prioritized soft goals that relate to system qualities, respectively to stakeholder needs, are candidates for inquiry of the corresponding Quality-Impact relationships. These candidates are used as an input to the tailoring of the Quality-Impact Inquiry method.

The Quality-Impact Inquiry method utilizes supporting elicitation methods (Pohl & Rupp, 2011), in particular the use of questionnaires, prototypes, and workshops. The method combines these supporting methods into a structured process for creating and analyzing evidence for decision-making about good-enough quality. Recommendations about good practice, e.g. of how to perform an effective workshop (Gottesdiener, 2002), should be followed as long as they do not interfere with the objective of the inquiry of Quality-Impact relationships that are under investigation. Side results from applying the method, e.g. the discovery of new needs or stakeholders during a workshop, should be embraced and handed-over as an input to the main stream of requirements engineering work that is performed in the development project.

In a larger scale validation of the proposed method in a real world situation various stakeholders and experienced requirements engineers are involved. To achieve trustworthy results, a specific probability is identified for considering a confident interval in which the value of quality impact lies within a specific range. Smaller numbers of stakeholders that involve in the experiment method generate wider confidence intervals since there is an inverse square root relationship between the confidence interval and the sample size. It means that to cut error margin in half, number of involved stakeholders is assumed to be four times more.

For practitioners, the Quality-Impact Inquiry method represents an extension of the requirements engineering toolset and is used for addressing the challenging problem of determining good-enough product quality. Once the relevant Quality-Impact relationships have been established, they can be reused while evolving and maintaining the application and for specifying the quality levels of comparable applications, for example in a software product line.

Quality-Impact Inquiry is not a method that is easy to apply and should thus be used by requirement engineers that are experienced in experimentation with end-users. In many practical situations, this is unproblematic. It is common to use experienced requirements engineers for critical tasks such as the development of service level agreements of software-based services (Marilly, Martinot, Papini, & Goderis, 2002).

The Quality-Impact Inquiry method complements competitive analysis of product quality (Regnell et al., 2008). It allows a definition of thresholds for useful quality and excessive quality based on evidence gathered by analyzing the perception of stakeholders. In the example of QoS and QoE, the requirements engineer determines the service quality threshold by translating quality of experience judgments with the experimentally determined Quality-Impact relationship. In the real-world example described in this paper, the former was quantified with software reaction time and the latter expressed with the Mean Opinion Score. The questionnaire in Fig 3 shows that the relationship can also be calculated for other impacts. For example, question 3 was used to collected data about the strategic positioning of the feature according to the Quper model (Regnell et al., 2008). Question 4 allowed collecting data about the risk of churn. Any prior knowledge about the nature of the relationship, e.g. as expressed by the exponential function in (Fiedler et al., 2010), reduces the need for measurements, thus reduces the effort of Quality-Impact inquiry.

For research, an understanding of the generic relationships between levels of more types of software quality and impact is urgently needed. These generic relationships reduce the need for experimentation during real-world requirements elicitation by pointing to the functions that should be used during Quality-Impact inquiry. The characterization of the generic relationship between QoS and QoE as an exponential function (Fiedler et al., 2010) is an example of the research that is needed. Security and usability are examples of quality attributes that should be prioritized by research. The research may include investigation of what appropriate measurement scales are, e.g. of security or usability, and how a generic Quality-Impact relationship may be expressed and investigated based on scales other than the ratio scale that we used in Fig 4 and Fig 5. Also open is the development of an understanding of how the interaction of multiple quality variables, e.g. security and usability (Braz et al., 2007), can be expressed with Quality-Impact relationships, thus made amenable to requirement elicitation with the Quality-Impact Inquiry method we have presented.

The study of Quality-Impact relationships would also allow building empirical evidence for checking deeply held beliefs in the requirements engineering field. One such belief is expressed with the KANO model (Sauerwein, Bailom, Matzler, & Hinterhuber, 1996). That model states that the impact of quality on stakeholder satisfaction is expressed through exponential or linear functions that describe attractive requirements, which cause delight when implemented, one-dimensional requirements, which are easily articulated, or must-be requirements, which are not obvious, but considered self-evident by stakeholders. The presented Quality-Impact Inquiry method enables practitioners to determine the exact relationships for the software products and features they are specifying. For researchers, it can be used to inform the design of empirical research studies that aim at investigating generic Quality-Impact relationships.

3.7 Conclusions

The paper has described an approach to quality requirements elicitation based on inquiry of Quality-Impact relationships. The method, called Quality-Impact Inquiry, guides a requirements engineer in the inquiry of good-enough software quality from the viewpoint of the appropriate stakeholders of the software system. When applying the method, stakeholders experience a prototype of a software system. The requirements engineer collects the real values of chosen quality attributes and subjective feedback from the stakeholders about perceived quality impacts. The analysis of Quality-Impact uses a regression function. The method can be tailored to pairs of qualities and impacts that are of interest for the specific software system. Systematic use of the method gives support for deciding about appropriate the quality levels. These can then be specified in a quantified manner for example by stating minimal, maximal, and expected quality in a software requirements specification (SRS) or service level agreement (SLA).

The Quality-Impact Inquiry method was applied for requirements engineering in real-world development projects. One example was shown to describe how to apply the method in practice and to report on lessons-learned. We reported how we have applied the method for these requirements engineering endeavors, shared early experiences from applying the method, and have given recommendations for practical use of the method.

Future research should aim at validating and evaluating the method in further, large-scale requirement engineering situations. Moreover, future research should aim at expanding the understanding of the generic relationships between given combinations of software quality attributes and their impacts as well as how quality attributes interact with each other. The resulting knowledge will translate into a SLA and help to allow and to reuse the knowledge of appropriate quality levels. It will also help accelerating and simplifying quality requirements inquiry in real-world projects, and enable research to check deeply held beliefs about how quality and impacts are interrelated.

Acknowledgments

This work has been co-sponsored by the European Commission through the FI-PPP integrated project FI-STAR under grant agreement number 318389.

CHAPTER 4 Quality of Experience Assessment Based on Analytics

Abstract

This work, which is connected to the Future Internet Public Private Partnership (FI-PPP) Integrated Project FI-STAR, presents a validation approach for Future Internet applications based on the use of analytics. In particular, it discusses how to use and combine software use and health statistics for the assessment of user-perceived Quality of Experience, in order to monitor user satisfaction, the risk of user churn, and the status of the corresponding ecosystem.

Keywords

QoE, QoS, response times, usage, analytics, churn, ecosystem

4.1 Introduction

Since more than a decade back in time, Quality of Experience (QoE) has become a key issue of concern for operators and providers, as bad QoE implies the risk of user churn (Le Callet et al., 2012). Indeed, when a service or application does not meet its stakeholder's expectations, economic loss is an almost unavoidable consequence. In particular, innovative applications are at risk once they do not succeed to satisfy their users.

In many sectors, the concern for quality has led to market entry barriers related to compliance, certification, and access to mission-critical data. In health and care, for example, IEC 80001 compliance, ISO 13485 certification, and access to data such as patient records are considered problematic (Thuemmler et al., 2013). Software product lines have been successfully used by companies to capture such domain-specific knowledge and thereby achieve systematic reuse across their product portfolio (Pohl, Böckle, & van der Linden, 2005). Such reuse is achieved in a software product line by engineering design specifications and components that embed commonality and variability across use cases of potential products. The impact is faster development and productization and better quality of applications and services.

The Future Internet Public Private Partnership (FI-PPP), a "European programme for Internet-enabled innovation", builds on this idea of product lines and attempts to scale it from a single product or services company to a whole industry (European-Commission, 2013). FI- PPP aims at establishing an evolving set of common components, called Generic Enablers (GE) that capture solutions to common problems in the building of internet- enabled applications and domain-specific problems such as interoperability with common devices and systems and those outlined above. The hope is that the resulting infrastructure advances the European markets for smart infrastructures, increases the effectiveness of business processes delivered through the Internet, and ultimately stimulates the economy.

In its first phase, a set of GEs have been developed, which aim at providing the basis for innovative applications in virtually any application domain (e.g. e-Health, logistics, energy, and etc.) within development cycles that are significantly shorter than those achieved so far. The GEs are offered by potentially competing manufacturers and producers. Application and service developers acquire these GEs for building applications in question.

The GE-based approach is comparable to buying the ingredients for a delicious home-prepared meal in a supermarket. Obviously, both the quality of the ingredients and their skillful preparation determine the quality of the prepared meal. The host can judge the quality of the meal by looking at its look, smell, and taste. The ultimate judgments of that quality, however, is seen in the appraisals of the host's guests and in the amount that people eat and are willing to return to eat upon the host's invitation. Translation of this metaphor to the domain of the Future Internet, makes it obvious that (1) the quality of the GEs and (2) the way these GEs are composed make a difference for a developed application as well as the corresponding ecosystem (Laghari & Connelly, 2012). The impact of these two concerns can be seen from (a) the comments of the users, and (b) the degree of usage.

How hosts, respectively product and service organizations, achieve good-enough quality throughout the whole value chain, from ingredients to the guests' experience and attitude, is the research underlying this paper. Our approach is based on the idea that the health of applications and their ingredients (such as GEs) needs to be measured, and that its impact on usage needs to be monitored, in order to be able to assure sufficient Quality of Experience.

The FI-PPP Integrated Project FI-STAR (FI-STAR, 2013) will address such validation, and develop and implement the corresponding measurement and analysis tools as follow-up of the ongoing requirement elicitation work. This paper reflects the approach to application and GE validation within FI-STAR and its seven use cases.

The remainder of the paper is structured as follows. Section II introduces an example of a FI-PPP based system and reviews existing work for quality evaluation of such system. Section III describes the analytics-based approach for QoE prediction and assessment. Section IV summarizes and concludes the paper with planned future work.

4.2 Background

Building a new system that meets its quality requirements is inherently difficult. Such requirements are often stated qualitatively like "the system must be fast", hence are ambiguous and thus difficult to verify (Glinz, 2008). When implementing such requirements the following kinds of problems may be encountered. Developers build a system that delivers less than the stakeholders expect. This results in stakeholder dissatisfaction and might render a system useless. Developers build a system that delivers more than the stakeholders need. This results in a system that is more expensive than necessary.

Quality is particularly important for heterogeneously sourced systems such as FI-PPP-based systems. When engineering such system, developers depend on components, applications, and services provided by third parties. Developers give such trust only if solution providers keep their promises regarding the service levels that will be achieved. Analytics provide transparency for evaluating such third-party contributions, for predicting the quality of the system, and for monitoring if the running service performs as promised. Analytics also provide the basis for root-cause analysis if quality objectives have not been met.

Figure 4-1 shows such a heterogeneously sourced system, a simplified and anonymized version of a FI-STAR use case scenario (www.fi-star.eu). The system allows patients and clinicians to collect and exchange biometric and other patient data. The system creates value by empowering the patient with rapid feedback about his condition and by providing treatment decision-support to the clinician.

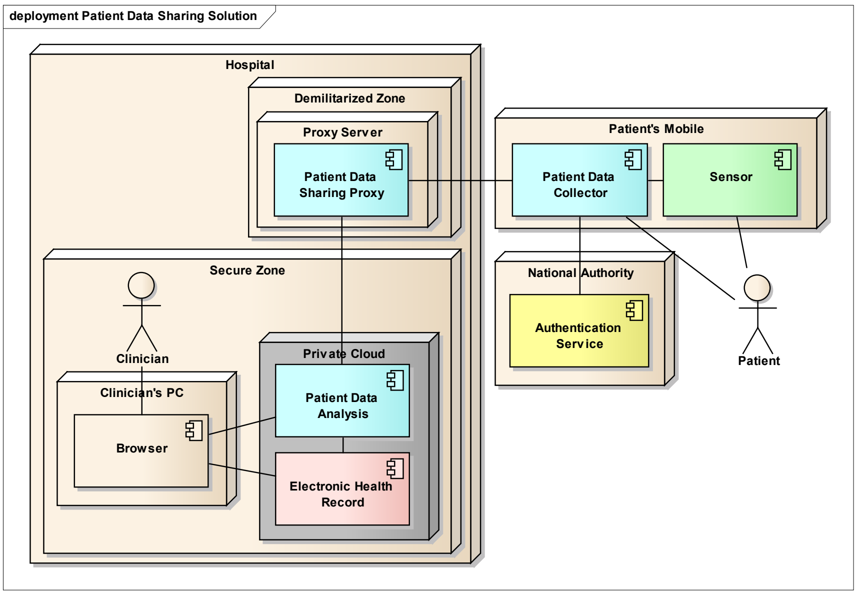


Figure 4-1: Patient data sharing solution. The letters in parentheses refer to suppliers the corresponding items are sourced from

According to the system architecture specification, the system consisted components, applications, devices, and services sourced from multiple parties. Patients would access the system with their personal mobile phone. The patient data collector, sharing proxy, and analysis applications would be developed by a software product company active in the healthcare domain. The sensors would be procured from a device manufacturer. User authentication services would be provided by the relevant national authority. The electronic health record would be managed by the hospital for which the solution was designed. The hospital-internal private cloud services, accessed by the clinician with one of the common web-browsers, would be provided by a local service provider. Components for connectivity and interoperability, finally, would be provided as GEs by FI-WARE platform providers.

A potentially wide variety of quality characteristics need to be fulfilled for given components, applications, and services to become useful. Such quality characteristics include functional suitability, performance, compatibility, usability, reliability, security, maintainability, and portability (ISO/IEC-25010, 2010). The quality levels achieved by a component or application is specified in the release requirements of that component or application. Warranties are used to guarantee that a product performs as promised in the specification. Usually, such a warranty is agreed between the supplier and the customer in a licensing contract (Kittlaus & Clough, 2009). Correspondingly, if a supplier provides a service for a customer, they agree on the quality of the service in a service level agreement (SLA). An SLA again specifies the quality levels, which the supplier gives warranty for. Norms, standards, and certificates are used to specify minimal quality levels to be achieved by products in a given industry (Thuemmler et al., 2013).

Once developed, integrated, and deployed, the quality of the system affects the quality of the user's experience (Fiedler et al., 2010). Quality can be so good that it allows the supplier to compete with alternative solutions (Regnell et al., 2008). If quality falls below the utility breakpoint, however, users will turn away and discard the solution (Khirman & Henriksen, 2002).

One approach to manage quality proactively is the use of software analytics (Menzies & Zimmermann, 2013). With analytics, attributes of software entities are measured, the measurements analysed and transformed into indicators that are useful for decision-making (ISO/IEC-15939, 2007). Such measurements give transparency, thus allows developers and management to decide about the course of actions for evolving the software system (Fotrousi et al., 2013).

A wide variety of analytics are used to manage the quality of the software engineering process, the quality of the resulting software products, and software systems that are in operation. Developer dashboards improve awareness of a project's situation to support planning and coordination (Baysal, Holmes, & Godfrey, 2013). Such dashboards include information about the organization, plans and tasks, source code and builds, and quality assurance (Czerwonka, Nagappan, Schulte, & Murphy, 2013). Prior to release, analytics allow analyzing performance and reliability of software and services (D. Zhang et al., 2013). Similar analytics and geo-location are used to monitor and improve performance of the service in a real-world context with the intended users (Musson et al., 2013). Voting buttons were proposed for measuring quality of experience. In comparison to laboratory testing, such late-stage analytics give diverse and representative results because they come from real use. Learning organizations use them to validate and improve testing assumptions.

Even-though analytics are effective for managing quality of software, their use is difficult to plan. In particular, it is unclear what an effective analytics approach is for managing quality when a heterogeneously sourced system such as the one outlined above is being developed. Too many variables could be measured, and trade-offs need to be made between ease of data collection and value of the analysis (Guest, Bunce, & Johnson, 2006). In addition, the composition of a system with multiple heterogeneous parts by one player and the use of the same part by different players makes standardization of a small set of broadly useful measurements important.

4.3 Approach

Our approach of predicting quality of experience (QoE) is based on three models: a measurement model, a composition model, and a lifecycle model.

The measurement model defines how quality attributes are measured and used to assert about properties of software or of users. It closely follows ISO/IEC 15939 for analytics measurement and ISO/IEC 25010 for quality attributes.

The software composition model defines how quality propagates as a result of composing software into real-world solutions. The approach follows the ideas of soft goal networks that allow deriving high-level global quality properties from low-level measurements (Chung et al., 2000; Haigh, 2010).

The software lifecycle model determines when measurements are made and quality assessed or predicted. It follows the principles of product management (Fricker, 2012), where the release of a software product is prepared, made available for customers, and integrated by such customers into larger solutions.

4.3.1 Measurement Model

The measurement model describes how data is collected to make assertions about quality of service and of experience. In our cooking metaphor, such data collection corresponds to the host that probes the ingredients or meal and interviews the guests. Probes include looking, taking a smell, and tasting the food and asking guests whether they like its appearance and taste. The host uses this data to understand whether the food meet the desired quality standards and to understand the guest's experience with it. Some of these properties can be derived from the corresponding measurement. For example, bad smell can be an indicator for bad food. Other properties can be inferred from indirect measurements. For example, whether the food was good can be inferred by asking the guests about their opinion. Similarly, experienced cooks are able to accurately predict the guest's experience based on the just tasting the food. The assessment of the ultimate success is different, though. As hosts, we would define it as whether the guests are eating or not. This can be assessed by observing whether the guests are eating or not.

Figure 4-2 illustrates the application of these measurements to software that is used by a human user. The human user corresponds to the guest, the software to the meal, and the host to the software provider. Software analytics are applied at the software, and empirical inquiries performed with users. Both of them allow collecting data for assessing quality of the software, quality of the user experience. Also, either of them also allows assessing the ultimate success of the software: whether it is used or not.

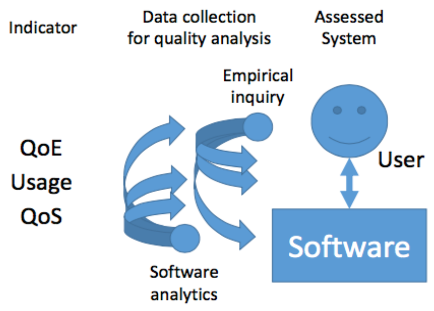


Figure 4-2: Measurement model: software analytics and empirical inquiry to assess QoS, QoE and usage of software

A substantial amount of work exists to understand how to assess software quality with analytics. Many address a selection of the software quality characteristics outlined by ISO/IEC 25010. The most common analytics are time and error-based.

Table 4-1: Measurement of software quality

	Time	**Error**	**OS/MOS**
Functional suitability			(Elbaum, Karre, & Rothermel, 2003; Kirakowski & Corbett, 1993; Lew, Olsina, & Zhang, 2010)
Performance	(Ran, 2003)		
Reliability	(Merzbacher & Patterson, 2002)	(Ran, 2003; Zheng & Lyu, 2010)	
Security	(Madan, Gogeva-Popstojanova, Vaidyanathan, & Trivedi, 2002)	(Madan et al., 2002)	
Usability			(Kirakowski & Corbett, 1993; Lew et al., 2010)

The most common empirical inquiry determines a score of user opinion. Table 4-1 gives an overview of existing work on how measurements are used to assess software quality. It excludes software qualities that affect stakeholders other than users. Table 4-1 illustrates the same idea: how measurements will be used to assess the impact of software on the user.

Evaluation of functional suitability of a software is usually performed by functional testing. However the result of functional suitability is reflected in terms of functional acceptability from user's perception. It can be reflected even in usage analytics (Elbaum et al., 2003). As an example, during a software use, unnecessary functions will be understood from click a stream that is an implication of functional inappropriateness.

Other aspects of software quality, usually called Quality of Service (QoS), are performance (Burby & Atchison, 2007) and reliability (Houtermans, Capelle, & Al-Ghumgham, 2007). QoS usually refers to system components and network delivery capacity. It concerns time behavior, resource utilization, and capacity aspects, in addition to availability, frequency of failures, fault tolerance rate, and recoverability time. Attributes such as throughput, loss ratio, jitter, packet error rate, response time, delay and availability time are vital for measuring in the network layer, and the transport layer between two machines (Hyun-Jong et al., 2008; Ran, 2003). Servers are measured by essential attributes of load rate, error rate, response time, peak response time, server up time, resource (i.e. CPU, memory, and disk) utilization, and threads (Bhatti & Friedrich, 1999). In the application layer, statistics about page errors, frame rate, call success rate and the quality of outputs such as audio, video, and files are identified to measure QoS (Mintauckis, 2010). Finally, security of an application/component affects the solution health (Hamam, Eid, El Saddik, & Georganas, 2008; Lindskog & Jonsson, 2002). The Attacks attribute is used to combat security issues such as DOS or malware attacks (Yadav & Gupta, 2013).

A time dependent attribute has the largest coverage for an end-to-end software health assessment. User perceived quality is dominated by response time and waiting time (Egger, Hossfeld, Schatz, & Fiedler, 2012; Xiong & Perros, 2009). The perception of quality on the user is typically measured by the Mean-Opinion-Score (MOS) (ITU-T, 2003). Availability of the software solution is measured by infinite response time. The response time of an intrusion tolerant system with the steady-state availability is monitored for the security assessment (Madan et al., 2002). Therefore response time can be a suitable candidate that simulates waiting time, availability as well as security. Error attribute provides further support for the assessment of software health in security, availability and fault tolerance.

Table 4-2: QoE measurements mapping to Quality in Use

	Time	**Error**	**OS/MOS**
Effectiveness			
Efficiency		↵	
Satisfaction			
Freedom from Risk			

4.3.2 Composition Model

The composition model describes how data is collected to combine assertions about quality of service and of experience. In our cooking metaphor, such composition corresponds to the host that combines and cooks the ingredient into a meal that is served to the guests. The host uses heating and combination to process the ingredients into a result of value higher for the guests than the inputs that were used. The quality of the inputs and the host's own work affect the quality of the results. The results are at most as good as the worst of the inputs that was used. Skilful preparation of the meal and presentation of it to the guests, however, can increase the value of the whole meal well beyond the sum of the inputs.

Figure 4-3.shows a software composition model that allows describing the solution shown in Figure 4-1. Nodes such as the private cloud contained in the secure zone, which again is contained in the hospital correspond to instances of the infrastructure. Patient Data Analysis and Electronic Health Record are two instances of software that run on the private cloud infrastructure. Not shown in Figure 4-1 are the generic enablers that the Patient Data Analysis contains. The clinician is a user that uses a browser, which communicates with the Patient Data Analysis and the Electronic Health Record software.

The composition model allows propagation of quality properties. Such propagation can be expressed in rules that are evaluated with an instance of the composition model (Figure 4-3 is such an instance). They determine how a property of one entity, for example a failure of an infrastructure, affects the rest of the software system. A set of availability-related rules would state that failure of infrastructure implies that any dependent software and user will experience the failure. Software that runs on reliable infrastructure, however, would not be affected by the failure. Similarly, a set of performance-related rules would state that the total round-trip time for a user interaction corresponds to the aggregated time behavior of software, run on the respective infrastructure, and communication channels. Depending on criticality of quality of service and experience, the set of rules can be completed and refined.

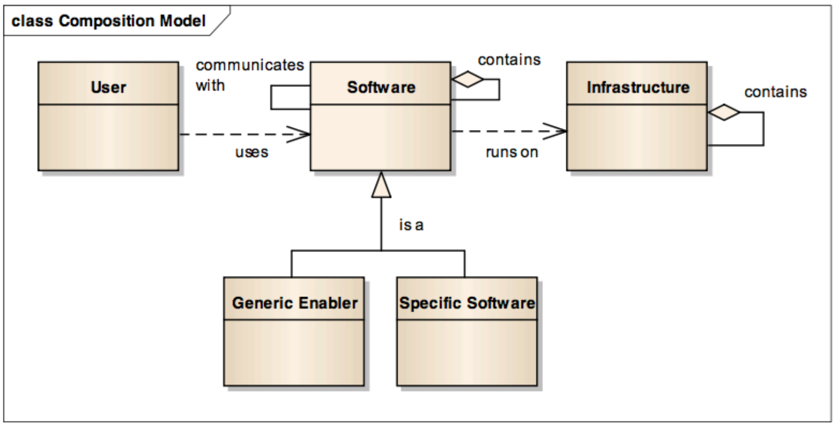


Figure 4-3: Composition model

4.3.3 Lifecycle Model

The lifecycle model describes how software and service infrastructure come into existence and evolve. The evolution stages then give raise to possible quality assurance actions. In our cooking metaphor, the cook would perform quality assurance actions based on the evolution stages of ingredients

and the meal. He would look for ingredients that are made available to him on the market. Preferably he would turn to ingredients with trusted quality, for example as indicated by certification labels awarded to some ingredients. In addition, he would touch and take a smell of some of them to assess their quality. Once in the kitchen, he would process and combine the ingredients into a meal. The meal undergoes quality assurance in the kitchen before it is made available to the guests. Once these guests have received the meal, they look at it and take a smell (presumably with delight) before they decide to eat it.

Figure 4-4 shows a lifecycle model that allows explaining how software is developed, delivered, integrated, and made available as a solution for the healthcare environment shown in Figure 4-1. Each supplier, indicated by the letter in parentheses in Figure 4-1, has developed, tested, and released software or infrastructure. The integrator then has performed acceptance testing of the sourced software and infrastructure in his own premises and integrated them into the solution that Figure 4-1 describes. Again the integrator tested and released the software solution, before performing site acceptance testing and initiating its usage.

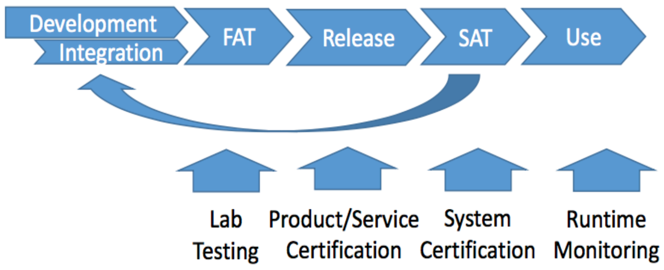


Figure 4-4: Software lifecycle model. FAT= factory acceptance test. SAT=site acceptance test.

The lifecycle model describes quality assurance actions that are performed at each respective lifecycle stage. Factory acceptance testing includes testing of the software in the supplier's laboratory environment. Software release is accompanied with certification. Such certification is standard practice of application stores such as Google Play and iTunes (Jansen & Bloemendal, 2013). Site acceptance testing is performed by the consumer of the released software in a laboratory environment that is as close to the real-world environment as possible. In the healthcare environment, site acceptance testing of software systems is accompanied IEC 80001 and ISO/IEC 27000 (Thuemmler et al., 2013). Systems that have passed all these quality assurance hurdles are put into use, where they continue to be monitored (Musson et al., 2013).

Each quality assurance action involves collection of analytics and possibly empirical data as described by the measurement model. The collected data updates earlier predictions made with the help of the composition model. Such updating allows validation of the prediction and increases confidence in whether the final solution actually meets its quality objectives or not.

The combination of the measurement and composition models enables early prediction. The lifecycle model allows planning for step-wise improvement of the prediction, hence reducing the risks of the final test of where a solution is being used in a real-world environment and the achieved quality of experience level determines success or failure of the system.

4.4 Conclusions

While important for any software, quality assurance is particularly critical for acceptance and successful use of heterogeneously sourced systems. Such systems integrate components from parties that the system integrator has little control over. As a consequence, the risk and the corresponding need for trust much higher than when a single-source software is developed.

This paper introduces a holistic approach for quality assurance of heterogeneously sourced systems. It is based on three models that together allow quality of experience prediction and step-wise validation of these predictions with real-world measurements. The measurement model describes how analytics and empirical data is collected and used for assertion of quality of service and experience. The composition model describes how measurements are propagated through the composed system to estimate overall quality of service and experience. The lifecycle model describes quality assurance actions that are used for validation of system quality.

The paper represents an important step towards unifying the so far separated disciplines of software engineering and performance evaluation in telecommunication systems. It contributes with a QoS and QoE measurement-based approach to managing quality while a software system is constructed. The paper explains the approach in depth with the metaphor of a host that prepares a delicious meal to guests. An exemplar taken from the FI-STAR project is taken to describe how the approach is transferred into a real-world environment.

Future work includes validation of the approach. Analysis of software architectures will be used for refining the composition model. A literature review will be performed for constructing a rule base for QoS and QoE assessment and prediction. Empirical inquiries about engineering process will be used to evaluate the composition model and refine the description of quality assurance practices. A particular focus will be given to the healthcare environment, where quality assurance is particularly important as it may decide on death or life.

Acknowledgment

This work has been co-sponsored by the European Commission, DG CNECT, through the FI-PPP Integrated Project FI-STAR.

Chapter 5 The Effect of Requests for User Feedback on Quality of Experience

[Since this chapter is submitted to a journal, it cannot be published online.]

References

Abelow, D. (1993). Automating feedback on software product use. *CASE Trends December*, 15-17.

Adamczyk, P. D., & Bailey, B. P. (2004). *If not now, when?: the effects of interruption at different moments within task execution*. Paper presented at the SIGCHI conference on Human factors in computing systems, Vienna, Austria.

Ahtinen, A., Mattila, E., Vaatanen, A., Hynninen, L., Salminen, J., Koskinen, E., & Laine, K. (2009). *User experiences of mobile wellness applications in health promotion: User study of Wellness Diary, Mobile Coach and SelfRelax*. Paper presented at the 3rd International Conference on Pervasive Computing Technologies for Healthcare, London, UK.

Antón, A. I., & Potts, C. (1998). *The use of goals to surface requirements for evolving systems.* Paper presented at the International Conference on Software Engineering, Kyoto, Japan.

Antons, J.-N., Arndt, S., Schleicher, R., & Möller, S. (2014). Brain activity correlates of quality of experience *Quality of Experience* (pp. 109-119): Springer.

Bailey, B. P., Konstan, J. A., & Carlis, J. V. (2001). *The effects of interruptions on task performance, annoyance, and anxiety in the user interface*. Paper presented at the IFIP International Conference on Human Computer Interaction (INTERACT), Tokyo, Japan.

Barbosa, O., & Alves, C. (2011). *A systematic mapping study on software ecosystems*. Paper presented at the The 2nd International Conference on Software Business (ICSOB 2011), Brussels, Belgium.

Barrett, L. F., Mesquita, B., & Gendron, M. (2011). Context in emotion perception. *Current Directions in Psychological Science, 20*(5), 286-290.

Bass, L., Clements, P., & Kazman, R. (2012). *Software Architecture in Practice* (3rd ed.): Addison-Wesley Professional.

Baysal, O., Holmes, R., & Godfrey, M. (2013). Developer Dashboards: The Need for Qualitative Analytics. *IEEE Software, 30*(4), 46-52.

Bevan, N. (1999). Quality in use: Meeting user needs for quality. *Journal of Systems and Software, 49*(1), 89-96.

Beyer, J., & Möller, S. (2014). Gaming *Quality of Experience* (pp. 367-381): Springer.

Bhatti, N. B., & Friedrich, R. (1999). Web server support for tiered services. *IEEE Network, 13*(5), 64-71. doi:10.1109/65.793694

Boegh, J. (2008). A New Standard for Quality Requirements. *IEEE Software, 25*(2), 57-63.

Boley, H., & Chang, E. (2007). Digital ecosystems: Principles and semantics.

Bosse, T., Broekens, J., Dias, J., & van der Zwaan, J. (2014). *Emotion Modeling: Towards Pragmatic Computational Models of Affective Processes*: Springer.

Braz, C., Seffa, A., & M'Raihi, D. (2007). *Designing a Trade-Off Between Usability and Security: A Metrics-Based Model.* Paper presented at the 11th IFIP TC 13 International Conference on Human-Computer Interaction (INTERACT 2007), Rio de Janeiro, Brazil.

Broekens, J., Pommeranz, A., Wiggers, P., & Jonker, C. M. (2010). *Factors influencing user motivation for giving online preference feedback.* Paper presented at the 5th Multidisciplinary Workshop on Advances in Preference Handling (MPREF'10), Lisbon, Portugal.

Brooks, P., & Hestnes, B. (2010). User measures of quality of experience: why being objective and quantitative is important. *Network, IEEE, 24*(2), 8-13. doi:10.1109/MNET.2010.5430138

Burby, J., & Atchison, S. (2007). *Actionable web analytics: using data to make smart business decisions*: Wiley. com.

Buse, R., & Zimmermann, T. (2010). *Analytics for software development.* Paper presented at the Foundations of Software Engineering (FSE)/SDP workshop on Future of software engineering research, Santa Fe, NM, USA.

Buse, R., & Zimmermann, T. (2012). *Information needs for software development analytics.* Paper presented at the Proceedings of the 2012 International Conference on Software Engineering.

Canale, S., Facchinei, F., Gambuti, R., Palagi, L., & Suraci, V. (2014). *User profile based Quality of Experience*. Paper presented at the 18th Internation Conference on Computers, Santorini Island, Greece.

Chapin, S. F., Torn, M. S., & Tateno, M. (1996). Principles of ecosystem sustainability. *American Naturalist*, 1016-1037.

Chung, L., Nixon, B., Yu, E., & Mylopoulos, J. (2000). *Non-Functional Requirements in Software Engineering* (Vol. 5). Boston, USA: Springer US.

Clifton, B. (2012). *Advanced web metrics with Google Analytics*: Wiley. com.

Cokins, G. (2009). *Performance management: Integrating strategy execution, methodologies, risk, and analytics* (Vol. 21): John Wiley & Sons.

Costanza, R. (1992). Toward an operational definition of ecosystem health. *Ecosystem health: New goals for environmental management*, 239-256.

Costanza, R., & Mageau, M. (1999). What is a healthy ecosystem? *Aquatic ecology, 33*(1), 105-115.

Côté, N., & Berger, J. (2014). Speech Communication *Quality of Experience* (pp. 165-177): Springer.

Creswell, J. W. (2014). *Research design: Qualitative, quantitative, and mixed methods approaches* (4 ed.): Sage publications.

Cysneiros, L. M., & Sampaio do Prado Leite, J. C. (2004). Nonfunctional requirements: From elicitation to conceptual models. *IEEE Transactions on Software Engineering, 30*(5), 328-350.

Czerwonka, J., Nagappan, N., Schulte, W., & Murphy, B. (2013). Codemine: Building a software development data analytics platform at microsoft. *Software, IEEE, 30*(4), 64-71.

Davenport, T. H., & Harris, J. G. (2007). *Competing on analytics: the new science of winning*: Harvard Business Press.

Delen, D., & Demirkan, H. (2013). Data, information and analytics as services. *Decision Support Systems, 55*(1), 359-363.

Doerr, J., Kerkow, D., Koenig, T., Olsson, T., & Suzuki, T. (2005). *Non-functional requirements in industry-three case studies adopting an experience-based NFR method.* Paper presented at the 13th IEEE International Conference on Requirements Engineering, Paris, France.

Egger, S., Hossfeld, T., Schatz, R., & Fiedler, M. (2012). *Waiting times in quality of experience for web based services.* Paper presented at the Fourth International Workshop on Quality of Multimedia Experience (QoMEX2012), Yarra Valley, Australia.

Elbaum, S., Karre, S., & Rothermel, G. (2003). *Improving web application testing with user session data*. Paper presented at the 25th International Conference on Software Engineering (ICSE'03), Portland, OR, USA.

Elo, S., & Kyngäs, H. (2008). The qualitative content analysis process. *Journal of advanced nursing, 62*(1), 107-115.

European-Commission. (2013). FI-PPP | Future Internet PPP. Retrieved from http://www.fi-ppp.eu

Feiten, B., Garcia, M.-N., Svensson, P., & Raake, A. (2014). Audio Transmission *Quality of Experience* (pp. 229-245): Springer.

FI-STAR. (2013). Home: FI-STAR. Retrieved from http://www.fi-star.eu

FI-STAR. (2015). Specific Enablers, Fi-Star Catatlouge. Retrieved from http://catalogue.fi-star.eu/enablers

FI-WARE. (2015). About FIWARE. Retrieved from https://www.fiware.org/about-us/

Fiedler, M., & Hoßfeld, T. (2010). *Quality of Experience-related differential equations and provisioning-delivery hysteresis.* Paper presented at the 21st ITC Specialist Seminar on Multimedia Applications-Traffic, Performance and QoE Miyazaki, Japan.

Fiedler, M., Hossfeld, T., & Tran-Gia, P. (2010). A Generic Quantitative Relationship between Quality of Experience and Quality of Service. *IEEE Network, 24*(2), 36-41.

Fotrousi, F. (2015). QoE-Probe Android. Retrieved from https://github.com/farnazfotrousi/QoE-Probe-Android

Fotrousi, F., Fricker, S. A., & Fiedler, M. (2014). *Quality requirements elicitation based on inquiry of quality-impact relationships*. Paper presented at the 22nd IEEE International Conference on Requirements Engineering, Karlskrona, Sweden.

Fotrousi, F., Izadyan, K., & Fricker, S. A. (2013). *Analytics for Product Planning: In-Depth Interview Study with SaaS Product Managers*. Paper presented at the IEEE 6th International Conference on Cloud Computing (Cloud 2013), Santa Clara, CA, USA.

Fricker, S. A. (2012). Software product management *Software for People* (pp. 53-81): Springer.

Fricker, S. A., & Glinz, M. (2010). *Comparison of Requirements Hand-Off, Analysis, and Negotiation: Case Study*. Paper presented at the 18th IEEE International Requirements Engineering Conference (RE'10), Sydney, Australia.

Fricker, S. A., Gorschek, T., Byman, C., & Schmidle, A. (2010). Handshaking with Implementation Proposals: Negotiating Requirements Understanding. *IEEE Software, 27*(2), 72-80.

Fricker, S. A., Schneider, K., Fotrousi, F., & Thuemmler, C. (2015). Workshop videos for requirements communication. *Requirements engineering*, 1-32. doi:10.1007/s00766-015-0231-5

Froehlich, J., Chen, M. Y., Consolvo, S., Harrison, B., & Landay, J. A. (2007). *MyExperience: a system for in situ tracing and capturing of user feedback on mobile phones*. Paper presented at the 5th international conference on Mobile systems, applications and services (MobiSys2007), San Juan, Puerto Rico.

Garcia, M.-N., Argyropoulos, S., Staelens, N., Naccari, M., Rios-Quintero, M., & Raake, A. (2014). Video Streaming *Quality of Experience* (pp. 277-297): Springer.

Gilb, T. (2005). *Competitive Engineering: A Handbood for Systems Engineering, Requirements Engineering, and Software Engineering using Planguage*: Butterworth-Heinemann.

Glinz, M. (2005). *Rethinking the Notion of Non-Functional Requirements*. Paper presented at the 3rd World Congress for Software Quality, Munich, Germany.

Glinz, M. (2007). *On Non-Functional Requirements*. Paper presented at the IEEE International Requirements Engineering Conference (RE'07), New Delhi, India.

Glinz, M. (2008). A Risk-Based, Value-Oriented Approach to Quality Requirements. *IEEE Software, 25*(2), 34-41.

Golafshani, N. (2003). Understanding reliability and validity in qualitative research. *The qualitative report, 8*(4), 597-606.

Golaszewski, S. (2013). Flexisketch. Retrieved from https://play.google.com/store/apps/details?id=ch.uzh.ifi.rerg.flexisketch&hl=en

Gottesdiener, E. (2002). *Requirements by Collaboration: Workshops for Defining Needs*: Addison-Wesley Professional.

Guest, G., Bunce, A., & Johnson, L. (2006). How many interviews are enough? An experiment with data saturation and variability. *Field methods, 18*(1), 59-82.

Haigh, M. (2010). Software quality, non-functional software requirements and IT-business alignment. *Software Quality Journal, 18*(3), 361-385.

Hamam, A., Eid, M., El Saddik, A., & Georganas, N. D. (2008). *A quality of experience model for haptic user interfaces*. Paper presented at the

Ambi-Sys workshop on Haptic user interfaces in ambient media systems (HAS 2008), Quebec City, Canada.

Hassenzahl, M., Wessler, R., & Hamborg, K.-C. (2001). *Exploring and understanding product qualities that users desire.* Paper presented at the 5th Annual Conference of the Human-Computer Interaction Group of the British Computer Society (IHm-HCI 01), Lille, France.

Herrera, M., Moraga, M. Á., Caballero, I., & Calero, C. (2010). Quality in use model for web portals (QiUWeP) *Current Trends in Web Engineering* (pp. 91-101): Springer.

Herrmann, A., & Paech, B. (2008). MOQARE: misuse-oriented quality requirements engineering. *Requirements engineering, 13*(1), 73-86.

Hoßfeld, T., Keimel, C., Hirth, M., Gardlo, B., Habigt, J., Diepold, K., & Tran-Gia, P. (2014). Best practices for QoE crowdtesting: QoE assessment with crowdsourcing. *IEEE Transactions on Multimedia, 16*(2), 541-558.

Houtermans, M., Capelle, T. V., & Al-Ghumgham, M. (2007). *Reliability Engineering & Data Collection*. Paper presented at the Second International Conference on Systems (ICONS'07), Martinique, France.

Hsieh, H.-F., & Shannon, S. E. (2005). Three approaches to qualitative content analysis. *Qualitative health research, 15*(9), 1277-1288.

Hyun-Jong, K., Dong Hyeon, L., Jong Min, L., Kyoung-Hee, L., Won, L., & Seong-Gon, C. (2008, 2-4 Sept. 2008). *The QoE Evaluation Method through the QoS-QoE Correlation Model.* Paper presented at the Fourth International Conference on Networked Computing and Advanced Information Management (NCM '08).

Iansiti, M., & Richards, G. L. (2006). Information Technology Ecosystem: Structure, Health, and Performance. *Antitrust Bull., 51*, 77.

IBosch, J. (2009). *From software product lines to software ecosystems*. Paper presented at the 13th International Software Product Line Conference (SPLC 2009), San Francisco, CA, USA.

Irvine, C., & Levin, T. (2000). *Quality of Security Service*. Paper presented at the 2000 Workshop on New Security Paradigms (NSPW'00), New York, NY, USA.

ISO/IEC-9126. (2001[part1] - 2003 [part2, part3]). ISO Standard 9126: Software Engineering – Product Quality. Geneve: International Organization for Standarization

ISO/IEC-15939. (2007). Systems and Software Engineering - Measurement Process (Vol. ISO/IEC 15939): International Organization for Standarization.

ISO/IEC-25010. (2010). Systems and Software Quality Requirements and Evaluation (Vol. ISO/IEC FDIS 25010): International Organization for Standarization.

ITU-T. (2003). ITU-T P.800. *in Mean Opinion Score(MOS) terminology, ed: Telecommunication Standardization Sector of ITU.*

Ivory, M. Y., & Hearst, M. A. (2001). The state of the art in automating usability evaluation of user interfaces. *ACM Computing Surveys (CSUR), 33*(4), 470-516.

Jacobs, S. (1999). *Introducing Measurable Quality Requirements: A Case Study*. Paper presented at the 4th IEEE International Symposium on Requirements Engineering (RE'99), Limerick, Ireland.

Jansen, S., & Bloemendal, E. (2013). Defining app stores: The role of curated marketplaces in software ecosystems *Software Business. From Physical Products to Software Services and Solutions* (pp. 195-206): Springer.

Jansen, S., Finkelstein, A., & Brinkkemper, S. (2009). *A sense of community: A research agenda for software ecosystems*. Paper presented at the 31st International Conference on Software Engineering (ICSE 2009) Vancouver, Canada.

Jung, G., Hiltunen, M., Joshi, K., Schlichting, R., & Pu, C. (2010). *Mistral: Dynamically Managing Power, Performance, and Adaptation Cost in Cloud Infrastructures*. Paper presented at the IEEE International Conference on Distributed Computing Systems (ICDCS 2010), Genoa, Italy.

Karapanos, E. (2013). User experience over time *Modeling Users' Experiences with Interactive Systems* (pp. 57-83): Springer.

Khan, A., Sun, L., Jammeh, E., & Ifeachor, E. (2010). Quality of experience-driven adaptation scheme for video applications over wireless networks. *IET communications, 4*(11), 1337-1347.

Khirman, S., & Henriksen, P. (2002). *Relationship between quality-of-service and quality-of-experience for public internet service*. Paper presented at the 3rd Workshop on Passive and Active Measurement, Fort Collins, CO, USA.

Kilkki, K. (2008). Quality of Experience in Communications Ecosystem. *Journal of Universal Computer Science, 14*(5), 615-624.

Kim, H.-J., Lee, D. H., Lee, J. M., Lee, K.-H., Lyu, W., & Choi, S.-G. (2008). *The QoE evaluation method through the QoS-QoE correlation model.* Paper presented at the Fourth International Conference on Networked Computing and Advanced Information Management (NCM'08) Gyeongju, Korea.

Kim, J. H., Gunn, D. V., Schuh, E., Phillips, B., Pagulayan, R. J., & Wixon, D. (2008). *Tracking real-time user experience (TRUE): a comprehensive instrumentation solution for complex systems*. Paper presented at the SIGCHI conference on Human Factors in Computing Systems, Florence, Italy.

Kirakowski, J., & Corbett, M. (1993). SUMI: The software usability measurement inventory. *British journal of educational technology, 24*(3), 210-212.

Kittlaus, H.-B., & Clough, P. (2009). *Software Product Management and Pricing*: Springer.

Klas, M., Heidrich, J., Munch, J., & Trendowicz, A. (2009). *CQML Scheme: A classification scheme for comprehensive quality model landscapes*. Paper presented at the 35th Euromicro Conference on Software Engineering and Advanced Applications (SEAA'09), Patras, Greece.

Kujala, S. (2003). User involvement: a review of the benefits and challenges. *Behaviour & information technology, 22*(1), 1-16.

Kujala, S., & Miron-Shatz, T. (2013). *Emotions, experiences and usability in real-life mobile phone use*. Paper presented at the SIGCHI Conference on Human Factors in Computing Systems, Paris, France.

Kusters, R. J., van Solingen, R., & Trienekens, J. J. (1999). Identifying embedded software quality: two approaches. *Quality and Reliability Engineering International, 15*(6), 485-492.

Laghari, K. U. R., & Connelly, K. (2012). Toward total quality of experience: A QoE model in a communication ecosystem. *Communications Magazine, IEEE, 50*(4), 58-65.

Le Callet, P., Möller, S., & Perkis, A. (2012). Qualinet white paper on definitions of quality of experience. *European Network on Quality of Experience in Multimedia Systems and Services*.

Lew, P., Olsina, L., & Zhang, L. (2010). *Quality, quality in use, actual usability and user experience as key drivers for web application evaluation* (Web Engineering ed.): Springer.

Lewis, M., Haviland-Jones, J. M., & Barrett, L. F. (2010). *Handbook of emotions*: Guilford Press.

Lindskog, S., & Jonsson, E. (2002). *Adding Security to Quality of Service Architectures.* Paper presented at the Proceedings of the SS-GRR Conference.

López, J. M., Fajardo, I., & Abascal, J. (2007). Towards Remote Empirical Evaluation of Web Pages' Usability *Human-Computer Interaction. Interaction Design and Usability* (pp. 594-603): Springer.

Madan, B. B., Gogeva-Popstojanova, K., Vaidyanathan, K., & Trivedi, K. S. (2002). *Modeling and quantification of security attributes of software systems*. Paper presented at the Dependable Systems and Networks (DSN 2002)

Manikas, K., & Hansen, K. M. (2013a). *Reviewing the Health of Software Ecosystems–A Conceptual Framework Proposal*. Paper presented at the International Workshop on Software Ecosystems (IWSECO 2013), Potsdam, Germany.

Manikas, K., & Hansen, K. M. (2013b). Software ecosystems–a systematic literature review. *Journal of Systems and Software, 86*(5), 1294-1306.

Marilly, E., Martinot, O., Papini, H., & Goderis, D. (2002). *Service level agreements: a main challenge for next generation networks.* Paper presented at the 2nd European Conference on Universal Multiservice Networks (ECUMN 2002) Colmar, France.

Menzies, T., & Zimmermann, T. (2013). Software analytics: so what? *IEEE Software, 30*(4), 31-37.

Merzbacher, M., & Patterson, D. (2002). *Measuring end-user availability on the web: Practical experience*. Paper presented at the Dependable Systems and Networks (DSN 2002), Bethesda, Maryland, USA.

Millon, T., Lerner, M. J., & Weiner, I. B. (2003). *Handbook of Psychology, Personality and Social Psychology* (Vol. 5): John Wiley & Sons.

Mills, A. J., Durepos, G., & Wiebe, E. (2009). *Encyclopedia of case study research* (Vol. 2): Sage Publications.

Minhas, T. N., & Fiedler, M. (2013). *Quality of experience hourglass model.* Paper presented at the International Conference on Computing, Management and Telecommunications (ComManTel), Ho Chi Minh City, Vietnam.

Mintauckis, K. (2010). *Empirical studies of Quality of Experience (QoE): A Systematic Literature Survey.* (Master of Science), University of OSLO.

Mitra, K., Zaslavsky, A., & Åhlund, C. (2011). *A probabilistic context-aware approach for quality of experience measurement in pervasive systems*. Paper presented at the 26th ACM symposium on applied computing, Taichung, Taiwan

Musson, R., Richards, J., Fisher, D., Bird, C., Bussone, B., & Ganguly, S. (2013). Leveraging the Crowd: How 48,000 Users Helped Improve Lync Performance. *IEEE Software, 30*(4), 38-45.

Pagano, D., & Brügge, B. (2013). *User involvement in software evolution practice: a case study.* Paper presented at the 35th international conference on Software engineering (ICSE 2013), San Francisco, CA, USA.

Parmenter, D. (2010). *Key performance indicators (KPI): developing, implementing, and using winning KPIs*: John Wiley & Sons.

Petersen, K., Feldt, R., Mujtaba, S., & Mattsson, M. (2008). *Systematic mapping studies in software engineering*. Paper presented at the 12th International Conference on Evaluation and Assessment in Software Engineering.

Pohl, K., Böckle, G., & van der Linden, F. J. (2005). *Software product line engineering: foundations, principles and techniques*: Springer Science & Business Media.

Pohl, K., & Rupp, C. (2011). *Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam - Foundation Level - IREB Compliant*: Rocky Nook Computing.

Potter, W. J., & Levine-Donnerstein, D. (1999). Rethinking validity and reliability in content analysis.

Potts, C., Takahashi, K., & Antón, A. I. (1994). Inquiry-based requirements analysis. *IEEE Software, 11*(2), 21-32.

Raake, A., & Egger, S. (2014). Quality and quality of experience *Quality of Experience* (pp. 11-33): Springer.

Ran, S. (2003). A model for web services discovery with QoS. *ACM Sigecom exchanges, 4*(1), 1-10.

Rapport, D. J., Costanza, R., & McMichael, A. J. (1998). Assessing ecosystem health. *Trends in Ecology & Evolution, 13*(10), 397-402.

Regnell, B., Berntsson Svensson, R., & Olsson, S. (2008). Supporting Roadmapping of Quality Requirements. *IEEE Software, 25*(2), 42-47.

Reiter, U., Brunnström, K., De Moor, K., Larabi, M.-C., Pereira, M., Pinheiro, A., . . . Zgank, A. (2014). Factors Influencing Quality of Experience *Quality of Experience* (pp. 55-72): Springer.

Rettig, M. (1994). Prototyping for Tiny Fingers. *Communications of the ACM, 37*(4), 21-27.

Roto, V., Law, E., Vermeeren, A., & Hoonhout, J. (2011). User Experience White Paper. *Bringing clarity to the concept of user experience*.

Santos, R., Werner, C. u., Barbosa, O., & Alves, C. (2012). *Software Ecosystems: Trends and Impacts on Software Engineering*. Paper presented at the 26th Brazilian Symposium in Software Engineering (SBES 2012).

Sauerwein, E., Bailom, F., Matzler, K., & Hinterhuber, H. H. (1996). *The Kano model: How to delight your customers.* Paper presented at the International Working Seminar on Production Economics, Igls, Innsbruck, Austria.

Scherer, K. R. (2005). What are emotions? And how can they be measured? *Social science information, 44*(4), 695-729.

Schleicher, R., Westermann, T., & Reichmuth, R. (2014). Mobile Human–Computer Interaction *Quality of Experience* (pp. 339-349): Springer.

Shaikh, J., Fiedler, M., & Collange, D. (2010). Quality of Experience from user and network perspectives. *annals of telecommunications-annales des telecommunications, 65*(1-2), 47-57.

Souag, A., Salinesi, C., & Wattiau, I. (2012). *Ontologies for Security Requirements: A Literature Survey and Classification*. Paper presented at the Advanced Information Systems Engineering Workshops, Gdańsk, Poland.

Strohmeier, D., Egger, S., Raake, A., Hoßfeld, T., & Schatz, R. (2014). Web Browsing *Quality of Experience* (pp. 329-338): Springer.

Szajna, B., & Scamell, R. W. (1993). The effects of information system user expectations on their performance and perceptions. *Mis Quarterly, 17*(4), 493-516. doi:10.2307/249589

Thomas, D. R. (2006). A general inductive approach for analyzing qualitative evaluation data. *American journal of evaluation, 27*(2), 237-246.

Thuemmler, C., Mival, O., Benyon, D., Buchanan, W., Paulin, A., Fricker, S., . . . Grottland, A. (2013). *Norms and standards in modular medical architectures.* Paper presented at the 15th International Conference on e-Health Networking, Applications & Services (Healthcom).

Varela, M., Skorin-Kapov, L., & Ebrahimi, T. (2014). Quality of Service Versus Quality of Experience *Quality of Experience* (pp. 85-96): Springer.

Wang, T., Si, Y., Xuan, X., Wang, X., Yang, X., Li, S., & Kavs, A. J. (2010). *A QoS ontology cooperated with feature models for non-functional requirements elicitation.* Paper presented at the Proceedings of the Second Asia-Pacific Symposium on Internetware, Suzhou, China.

Wechsung, I., & De Moor, K. (2014). Quality of Experience Versus User Experience *Quality of Experience* (pp. 35-54): Springer.

Weiblen, T., Giessmann, A., Bonakdar, A., & Eisert, U. (2012). *Leveraging the Software Ecosystem-Towards a Business Model Framework for Marketplaces*. Paper presented at the Dcnet/ice-b/optics.

Westin, S. S. (1998). *Performance measurement and evaluation: definitions and relationships*. Retrieved from http://www.gao.gov/assets/80/77277.pdf

Wieringa, R., Maiden, N., Mead, N., & Rolland, C. (2006). Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requirements engineering, 11*(1), 102-107.

Xiong, K., & Perros, H. (2009). *Service performance and analysis in cloud computing*. Paper presented at the World Conference on Services, Los Angeles, California, USA.

Yadav, P., & Gupta, G. (2013). Depleting Clouds. *International Journal of Engineering, 2*(4).

Yin, R. K. (2014). *Case study research: Design and methods* (5 ed.): Sage publications.

Zahariadis, T., Papadimitriou, D., Tschofenig, H., Haller, S., Daras, P., Stamoulis, G. D., & Hauswirth, M. (2011). *Towards a future internet architecture*: Springer.

Zhang, D., Dang, Y., Lou, J.-G., Han, S., Zhang, H., & Xie, T. (2011). *Software analytics as a learning case in practice: Approaches and experiences*. Paper presented at the Proceedings of the International Workshop on Machine Learning Technologies in Software Engineering.

Zhang, D., Han, S., Dang, Y., Lou, J.-G., Zhang, H., & Xie, T. (2013). Software Analytics in Practice. *IEEE Software, 30*(5), 30-37.

Zhang, J., & Ansari, N. (2011). On assuring end-to-end QoE in next generation networks: challenges and a possible solution. *IEEE Communications Magazine, 49*(7), 185-191.

Zheng, Z., & Lyu, M. R. (2010). An adaptive QoS-aware fault tolerance strategy for web services. *Empirical Software Engineering, 15*(4), 323-345.

Acronynms

FI-PPP: Future Internet Public Private Partnership

FI-STAR: Future Internet Social and Technological Alignment Research

GE: Generic Enabler

HCI: Human Computer Interaction

KPI: Key Performance Indicator

QoE: Quality of Experience

QoS: Quality of Service

MOS: Mean Opinion Score

NF: Non-Functional

OBJ: Objective

OS: Opinion Score

RQ: Research Question

SECO: Software ECOsystem

SLA: Service Level Agreement

SRS: Software Requirement Specification

UX: User Experience