Quality Requirements Elicitation based on Inquiry of Quality-Impact Relationships

Farnaz Fotrousi, Samuel Fricker, Markus Fiedler

22nd IEEE International Requirements Engineering Conference

2014 Karlskrona

Quality Requirements Elicitation Based on Inquiry of Quality-Impact Relationships

Farnaz Fotrousi
Blekinge Institute of Technology (BTH)
Karlskrona, Sweden
farnaz.fotrousi@bth.se

Samuel A. Fricker
Blekinge Institute of Technology (BTH)
Karlskrona, Sweden
samuel.fricker@bth.se

Markus Fiedler
Blekinge Institute of Technology (BTH)
Karlskrona, Sweden
markus.fiedler@bth

***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.**

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

I. Introduction

Quality requirements are an important class of non-functional requirements [1]. They concern software system attributes such as functional suitability, performance, reliability, usability, security, and portability that are important for achieving stakeholder goals [2]. 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 [3, 4].

Meeting the right level of quality is important to balance benefits and cost [5]. 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 [6]. Excessive quality may lead to an unnecessarily expensive design of the software system [7], to unnecessary consumption of resources needed for operating the system [8], and to trade-offs where other quality attributes suffer [9].

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 [10].

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.

II. 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 [11], thus called soft requirements [12].

The softness characteristic of implies that the right level of desired requirements quality needs to be identified during requirements engineering [5]. 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 [13, 14].

Goal models have been proposed to elicit quality requirements [3, 15]. 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 [16, 17]. The means-ends relationships that are an inherent part of a goal model make the impact of such a quality requirements explicit [18, 19]. 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 [3]. 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 [13].

Several supporting elicitation methods have been proposed for requirements elicitation [20]. 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 [21]. 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 [22, 23].

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 [24, 25] 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 [26]. 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 [27].

For a telecommunication system, Quality of Service (QoS) requirements are stated that concern system performance, availability, and capacity [17]. Often these requirements are agreed in a service level agreements (SLA) between the system customers and the supplier [28]. 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" [29]. It has been shown that a system's Quality of Service affects the user's Quality of experience [27]. 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.

III. 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 [30]. 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 [10]. These principles have been translated into a software requirements engineering context by integrating it into an inquiry-based requirements analysis process [31] 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 [31] 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.

Fig 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.

A. Inquiry Process

Fig 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 [32] 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

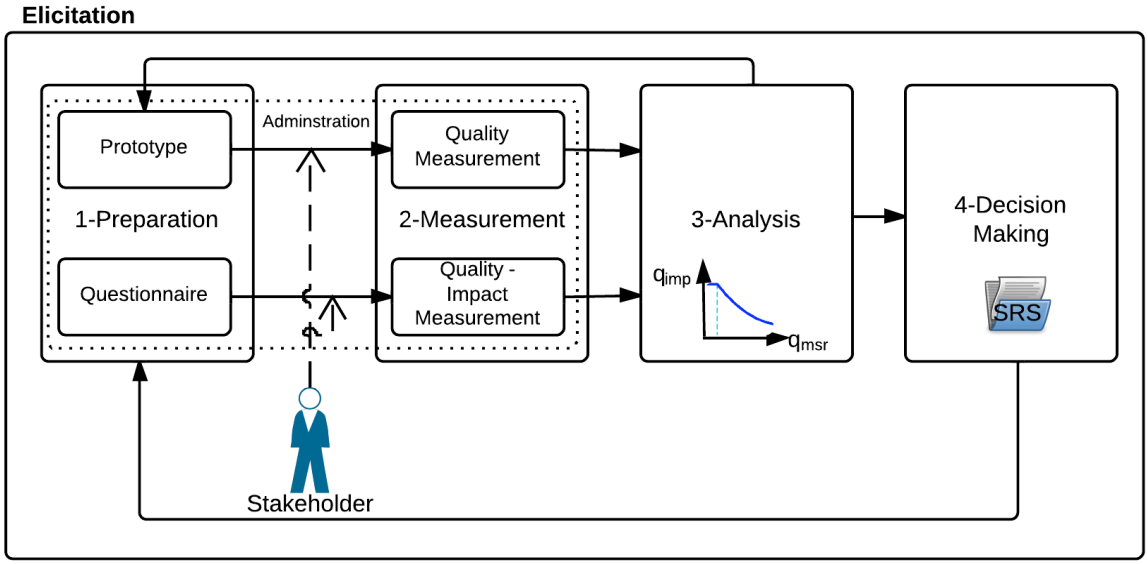

Fig 1. Quality-Impact Inquiry Method

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*:

$$U = \{u\} \tag{2}$$

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

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

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 [33]. 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,

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

and rationales in forms of comments that explain their ratings:

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

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

$$Q_{inUse} \subset Q \tag{6}$$

The quality impact is translated into a discrete value that is scaled using scores such as Mean Opinion Score (MOS) [34].

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 [35, 36]. The regression function is calculated for a feature *f* and quality attribute q_{att}:

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

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 [27].

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

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

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 Fig 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.

$$q_{new} = \{ g(\hat{q}_{msr}, q_{SRS}, comm , Q_{inUse} | f, q_{att}) \} \quad (9)$$

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

B. 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 1 gives an overview.

TABLE 1. Estimated quality values for given quality impacts

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.
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.

IV. REAL-WORLD EXAMPLE OF METHOD APPLICATION

A. 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.

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. Fig 2 gives an impression of the setup.

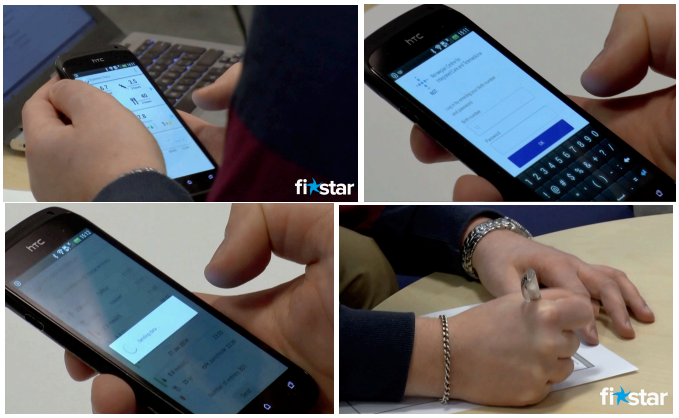

Fig 2: User interaction scenario with instrumented application and subsequent answering of the quality of experience questionnaire.

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. Fig 3 shows the questionnaire.

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:

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

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 Fig 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.

```
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
```

Fig 4: Extract from the log file with timestamps and activities

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.

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 [37]. 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:

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

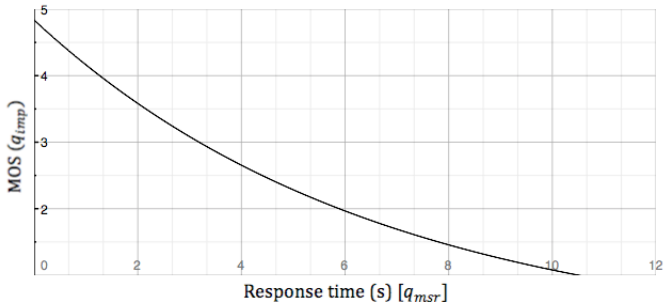

Fig 5. Quality impact (MOS) as a function of quality value (response time(s))

Fig 5 plots this regression function that shows quality impact (q_{imp}) as a function of quality value (q_{msr})[37]. The response time collected from different experiments as well as collected relevant quality impacts will plot the Fig 4. Taking the reverse of this function estimates quality value ($\hat{q}_{msr}$) as a function of quality impact (q_{imp}):

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

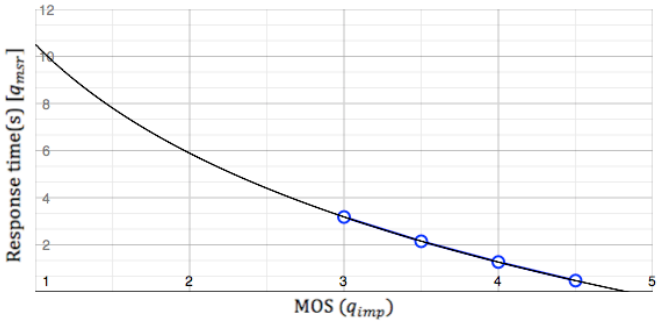

Fig 6. Quality value (Response time (s)) as a function of quality impact (MOS)

As Fig 6 plots the inverse regression function, shorter response identifies the better perception of quality and user's score. Table 2 estimates quality values for q_{imp} 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 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 [38].

B. Lesson learned

1) 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.

2) 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.

3) 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.

4) 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.

5) 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.

V. 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 [5]. 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 [3, 15]. 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 [20], 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 [39], 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 [40].

The Quality-Impact Inquiry method complements competitive analysis of product quality [5]. 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 [5]. 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 [10], 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 [10] 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 [9], 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 [41]. 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.

VI. Summary and 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.

VII. Acknowledgments

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

VIII. References

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

[2] J. Boegh, "A New Standard for Quality Requirements," IEEE Software, vol. 25, pp. 57-63, 2008.

[3] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, Non-Functional Requirements in Software Engineering. Boston, USA: Kluwer Academic Publishers, 2000.

[4] M. Haigh, "Software Quality, Non-Functional Software Requirements and IT-Business Alignment," Software Quality Journal, vol. 18, pp. 361-385, 2010.

[5] B. Regnell, R. Berntsson Svensson, and S. Olsson, "Supporting Roadmapping of Quality Requirements," IEEE Software, vol. 25, pp. 42-47, 2008.

[6] K. Kilkki, "Quality of Experience in Communications Ecosystem," Journal of Universal Computer Science, vol. 14, pp. 615-624, 2008.

[7] L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice, 3rd ed.: Addison-Wesley Professional, 2012.

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

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

[10] M. Fiedler, T. Hossfeld, and T.-G. Phuoc, "A Generic Quantitative Relationship between Quality of Experience and Quality of Service," IEEE Network, vol. 24, pp. 36-41, 2010.

[11] M. Glinz, "Rethinking the Notion of Non-Functional Requirements," presented at the 3rd World Congress for Software Quality, Munich, Germany, 2005.

[12] C. Irvine and T. Levin, "Quality of Security Service," presented at the 2000 Workshop on New Security Paradigms (NSPW'00), New York, NY, USA, 2000.

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

[14] T. Gilb, Competitive Engineering: A Handbood for Systems Engineering, Requirements Engineering, and Software Engineering using Planguage: Butterworth-Heinemann, 2005.

[15] A. I. Antón and C. Potts, "The use of goals to surface requirements for evolving systems," in International Conference on Software Engineering, Kyoto, Japan, 1998, pp. 157-166.

[16] A. Souag, C. Salinesi, and I. Wattiau, "Ontologies for Security Requirements: A Literature Survey and Classification," presented at the Advanced Information Systems Engineering Workshops, Gdańsk, Poland, 2012.

[17] T. Wang, Y. Si, X. Xuan, X. Wang, X. Yang, S. Li, et al., "A QoS ontology cooperated with feature models for non-functional requirements elicitation," in Proceedings of the Second Asia-Pacific Symposium on Internetware, Suzhou, China, 2010, p. 17.

[18] L. M. Cysneiros and J. C. Sampaio do Prado Leite, "Nonfunctional requirements: From elicitation to conceptual models," IEEE Transactions on Software Engineering, vol. 30, pp. 328-350, 2004.

[19] A. Herrmann and B. Paech, "MOQARE: misuse-oriented quality requirements engineering," Requirements Engineering, vol. 13, pp. 73-86, 2008.

[20] K. Pohl and C. Rupp, Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam - Foundation Level - IREB Compliant: Rocky Nook Computing, 2011.

[21] M. Rettig, "Prototyping for Tiny Fingers," Communications of the ACM, vol. 37, pp. 21-27, 1994.

[22] S. Fricker, T. Gorschek, C. Byman, and A. Schmidle, "Handshaking with Implementation Proposals: Negotiating Requirements Understanding," IEEE Software, vol. 27, pp. 72-80, 2010.

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

[24] M. Hassenzahl, R. Wessler, and K.-C. Hamborg, "Exploring and understanding product qualities that users desire," in 5th Annual Conference of the Human-Computer Interaction Group of the British Computer Society (IHm-HCI 01), Lille, France, 2001, pp. 95-96.

[25] R. J. Kusters, R. van Solingen, and J. J. Trienekens, "Identifying embedded software quality: two approaches," Quality and Reliability Engineering International, vol. 15, pp. 485-492, 1999.

[26] J. Doerr, D. Kerkow, T. Koenig, T. Olsson, and T. Suzuki, "Non-functional requirements in industry-three case studies adopting an experience-based NFR method," in 13th IEEE International Conference on Requirements Engineering, Paris, France, 2005, pp. 373-382.
[27] M. Fiedler, T. Hossfeld, and P. Tran-Gia, "A generic quantitative relationship between quality of experience and quality of service," Network, IEEE, vol. 24, pp. 36-41, 2010.
[28] H.-B. Kittlaus and P. Clough, Software Product Management and Pricing: Springer, 2009.
[29] P. Le Callet, S. Möller, and A. Perkis, "Qualinet White Paper on Definitions of Quality of Experience (2012)," European Network on Quality of Experience in Multimedia Systems and Services, Lausanne, SwitzerlandMarch 2013 2013.
[30] R. Wieringa, N. Maiden, N. Mead, and C. Rolland, "Requirements engineering paper classification and evaluation criteria: a proposal and a discussion," Requirements Engineering, vol. 11, pp. 102-107, 2006.
[31] C. Potts, K. Takahashi, and A. I. Antón, "Inquiry-based requirements analysis," IEEE software, vol. 11, pp. 21-32, 1994.
[32] P. Brooks and B. Hestnes, "User measures of quality of experience: why being objective and quantitative is important," Network, IEEE, vol. 24, pp. 8-13, 2010.
[33] M. Fiedler and T. Hoßfeld, "Quality of Experience-related differential equations and provisioning-delivery hysteresis," in 21st ITC Specialist Seminar on Multimedia Applications-Traffic, Performance and QoE Miyazaki, Japan, 2010.
[34] ITU, "ITU-T P.800," in Mean Opinion Score(MOS) terminology, ed: TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU, 2003.
[35] H.-J. Kim, D. H. Lee, J. M. Lee, K.-H. Lee, W. Lyu, and S.-G. Choi, "The QoE evaluation method through the QoS-QoE correlation model," in Fourth International Conference on Networked Computing and Advanced Information Management (NCM'08) Gyeongju, Korea, 2008, pp. 719-725.
[36] T. N. Minhas and M. Fiedler, "Quality of experience hourglass model," in International Conference on Computing, Management and Telecommunications (ComManTel), Ho Chi Minh City, Vietnam, 2013, pp. 87-92.
[37] J. Shaikh, M. Fiedler, and D. Collange, "Quality of Experience from user and network perspectives," annals of telecommunications-annales des telecommunications, vol. 65, pp. 47-57, 2010.
[38] A. Khan, L. Sun, E. Jammeh, and E. Ifeachor, "Quality of experience-driven adaptation scheme for video applications over wireless networks," IET communications, vol. 4, pp. 1337-1347, 2010.
[39] E. Gottesdiener, Requirements by Collaboration: Workshops for Defining Needs: Addison-Wesley Professional, 2002.
[40] E. Marilly, O. Martinot, H. Papini, and D. Goderis, "Service level agreements: a main challenge for next generation networks," in 2nd European Conference on Universal Multiservice Networks (ECUMN) Colmar, France, 2002, pp. 297-304.
[41] E. Sauerwein, F. Bailom, K. Matzler, and H. H. Hinterhuber, "The Kano model: How to delight your customers," in International Working Seminar on Production Economics, Igls, Innsbruck, Austria, 1996, pp. 313-327.