Stakeholder Involvement into Quality Definition and Evaluation for Service-Oriented Systems

Vladimir A. Shekhovtsov, Heinrich C. Mayr, Christian Kop
Application Engineering Group, Institute of Applied Informatics,
Alpen-Adria-Universität Klagenfurt, Klagenfurt, Austria
shekvl@yahoo.com, {heinrich, chris}@ifit.uni-klu.ac.at

***Abstract*—The paper addresses the matter of quality in the software process for service-oriented systems. We argue for the need of involving the users/stakeholders into the specification and evaluation of quality (requirements) and we develop means for supporting such an involvement. For this purpose we introduce classifications of user and quality types and as a basis for the characterization of evaluation cases.**

***Keywords*-business stakeholder involvement; quality assessment; software process; empirical evidence; service-oriented systems**

I. Introduction

Software processes for service-oriented systems cannot be successful without involving the respective business stakeholders. One important category of such involvement concerns *collecting stakeholder opinions and expectations on quality of the prospective system*: If the stakeholders do not have a chance to bring in their quality expectations early in the software development lifecycle, these can be easily lost - leaving the stakeholders dissatisfied late in the development lifecycle when it is costly to fix.

Organizing and supporting such kind of involvement still remains an open task for service-oriented systems:

1. It is difficult for stakeholders without IT experience to express their expectations on the quality of the system under development (SUD) if they cannot experience it in the appropriate context; without such experience, they are forced to be rough in their opinions formulating them as e.g. “the services must be reliable” etc..
2. The process of assessing service quality needs to be performed for different service compositions and for different cases of anticipated or implemented interaction of the services with the environment; this is a complex task as it is difficult to invent the service compositions and interaction scenarios “on the fly” without necessary support.
3. We know of no established process and tool support of quality-related interaction between business stakeholders and IT people relying on established common vocabulary.

In this paper, we address the need for facilitating the user evaluations [1] as a part of the approach to address the above problems. This approach is planned to be developed as a part of the QUASE-IOS project (short for QUality-Aware Software Engineering for the Internet of Services) established in cooperation with two local software development companies.

The paper is structured as follows. Section 2 describes the QUASE-IOS project. Section 3 introduces the cases for QUASE-IOS user evaluation activities, lists their attributes, and outlines the directions for the possible discussions related to such activities; it is followed by conclusions.

II. QUASE-IOS Project Description

The QUASE-IOS project (its topic was introduced in [2, 3]) aims at acquiring and formalizing knowledge about the perception and assessment of software quality by business stakeholders. This should facilitate the quality-related involvement of stakeholders into software processes targeting service-oriented systems.

Consequently, the project focuses on the following fields of research and development activities: ontological engineering, continuous quality awareness, model-based process support, collecting stakeholder evidence and forming quality awareness data.

Ontological engineering activities consist in acquiring and organizing the knowledge on quality-related software process activities into a common ontology (the Ontology of Stakeholder Quality Perception and Assessment, StakeQPA). In particular, that ontology has to incorporate the knowledge about software quality and its perception by stakeholders and IT people, the processes of stakeholder quality assessment, the ways of producing quality to be proposed to stakeholders, quality-observing contexts etc.. Thus it plays a central role when following the Ontology-based Software Engineering paradigm [4, 5].

Continuous quality awareness is defined as the software engineers’ ability to check, throughout the software process, the desired quality of the SUD against the current realization stage (Fig.1). To facilitate such awareness, we need to formalize the set of *awareness support processes* to be performed continuously on different software process stages (Fig.2). These processes should provide the means for getting the stakeholders’ quality expectations and forming the awareness support data.

In particular:

1. Service quality values should be formed in a way accessible to business stakeholders (e.g. simulated) and presented to them interactively in the context of the system usage;
2. Stakeholders should experience service qualities in usage contexts corresponding to their roles, and assess this experience using the specific scale;
3. These assessments incorporate software process activities like non-functional requirements (NFR) elicitation or the evaluation of design decisions.

For the determination of processes for continuous quality awareness we distinguish between *expected* and *proposed quality*.

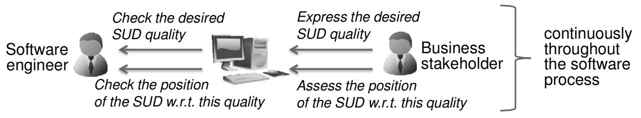
Check the desired SUD quality
Express the desired SUD quality
Software engineer
Business stakeholder
continuously throughout the software process
Check the position of the SUD w.r.t. this quality
Assess the position of the SUD w.r.t. this quality

Figure 1. Continuous quality awareness

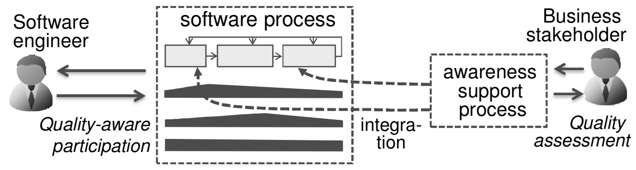

Figure 2. Integrating the awareness support into the software process

The *expected quality (EQ)* is defined as a quality perception a business stakeholder wants the final SUD version to provide. It needs to be considered whenever the software process deals with stakeholder opinions, but it exists only in a stakeholder's mind. Despite this "ephemeral" character, it has an important influence on the software process. For example, such quality indirectly but significantly affects stakeholders' decisions on the outcome of the whole project or on its milestones.

The *proposed quality (PQ)* denotes a set of values for quantified SUD quality attributes reflecting the current state of SUD development and represented in a form suitable for the perception by stakeholders. The s is expected to have that quality if implemented exactly as defined at the given development stage. It can be perceived by stakeholders both directly (via their senses) and indirectly (e.g. through the verbal description). There are several ways to produce proposed quality: from a (e.g. simulation) model, from a system prototype, from the complete system version.

Model-based process support follows the situational method engineering paradigm [5] and aims at composing a QUASE-IOS runtime model QRM as the backbone of a common QUASE-IOS runtime process QRP for supporting quality-related decisions. I.e., launching QRP means to enact QRM.

In this paper, we introduce the prospective user evaluations in the context of process support for involvement-related tasks.

Fig.3 depicts the QUASE-IOS runtime process First, the appropriate awareness support activity such as SUD state evaluation or NFR elicitation (see below) is invoked. In parallel, the software engineer is requested to provide the necessary information (such as the values of quality-influencing factors). When the values are set, the system prepares itself to execute the EQ collecting activity by launching the EQ collecting process, which incorporates the stakeholders, too.

To reveal EQ in contexts, in [2, 3] we proposed to represent such contexts by an interactive *usage process* defining the sequences of actions for stakeholder interaction sessions. Every such process defines the particular usage context and is presented by the EQ forming process to the stakeholder belonging to his/her particular role. Its input is the set of values for all the SUD services.

The outputs of these sub-processes include the set of all PQ values that are relevant for the particular context and the set of the corresponding EQ values revealed by the participating stakeholders.

Whilst a context-level usage process is performed *service-level revealing processes* handle the forming of PQ (e.g. by simulation) and the revealing of EQ (e.g. by assessment).. Their inputs are the values of the factors influencing the SUD quality (e.g. the expected user load).

An *EQ revealing process* organizes the interactions with stakeholders to reveal the particular EQ characteristics. It can incorporate human-computer interaction, human-to-human interaction (e.g. the analyst delivers the PQ value to the stakeholder and asks for assessment), and empirical method-based interaction (using questionnaires etc.).

A *PQ forming process* is to present the corresponding PQ to a revealed EQ. The particular implementation depends on the given quality characteristic and project type. Its input is a set of values for PQ-influencing factors. We plan to establish two categories of such processes: backed by real software solutions and based on simulation models for SUD qualities.

Forming the awareness data is based on the collection of both PQ values and their corresponding EQ values, from the results of the revealing interactions. If the ideal case, these values are obtained for all key usage contexts of every SUD service.

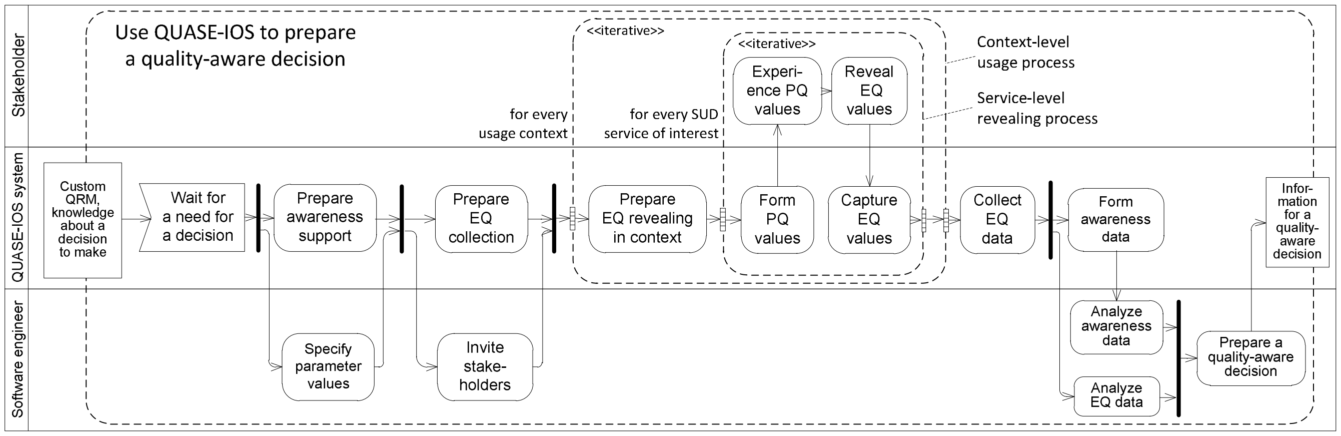
Stakeholder
QUASE-IOS system
Software engineer
Use QUASE-IOS to prepare a quality-aware decision
<<iterative>>
<<iterative>>
Experi-ence PQ values
Reveal EQ values
Context-level usage process
Service-level revealing process
for every usage context
for every SUD service of interest
Custom QRM, knowledge about a decision to make
Wait for a need for a decision
Prepare awareness support
Prepare EQ collection
Prepare EQ revealing in context
Form PQ values
Capture EQ values
Collect EQ data
Form awareness data
Infor-mation for a quality-aware decision
Specify parameter values
Invite stake-holders
Analyze awareness data
Analyze EQ data
Prepare a quality-aware decision

Figure 3. QUASE-IOS runtime process QRP

Quality awareness data are formed by appropriate support processes: the *non-functional requirements elicitation process* derives NFR thresholds out of the collected expected quality data, whereas the *SUD state evaluation process* converts the information about the design decision into the PQ data (e.g. via modifying the values for quality-influencing factors as a result of the architectural decision) and performs the assessment of the design decision via question-answering on the collected expected quality data.

This allows the software engineers to make quality-aware decisions. For a particular decision, the stakeholders can be asked to reveal the expected quality corresponding to the proposed quality formed to reflect the impact of that decision: A low assessment mark may indicate a problem with that decision.

III. User Evaluations in QUASE-IOS

In this section, we deal with the user evaluations in the QUASE-IOS project framework and discuss the relevant cases, the attributes of evaluation cases, and some challenges and possible directions for a further discussion.

We distinguish two main categories of user evaluations in the framework of the QUASE-IOS project: *internal* and *extern*al evaluations (Table 1).

Internal evaluation (*assessment*) is performed as a part of the QUASE-IOS runtime process. There is only one case of such evaluation, i.e. *stakeholder-driven internal evaluation (SIE):* a business stakeholder assesses proposed quality values for SUD services during the EQ revealing process as described above.

External evaluation is performed outside of the QUASE-IOS runtime process. We distinguish the following cases of external evaluations:

SEE - Stakeholder-driven external evaluation (a "meta-process"): a business stakeholder evaluates (at design time) the quality of the process of assessing the p values (i.e. actually, the quality of organizing the *SIE* evaluation in the EQ revealing process at runtime). Evaluations of that type support revealing the information about stakeholder quality perception and assessment to be integrated into the implementation of the EQ revealing processes; and to make the proposed solution meet the quality (e.g. usability) criteria from the point of view of the business stakeholders.

OEE - Ontological external evaluation: both IT people and business stakeholders evaluate the knowledge incorporated into the StakeQPA ontology. We apply for that evaluation established approaches (e.g. [6]).

CEE - Customer-driven external evaluation: IT people as the expected users of the QUASE-IOS software solution evaluate the quality of its implementation.

For these evaluation cases we can now distinguish the following attributes (see Table 1):

1. *Evaluation subject:* the category of users performing the evaluation; the most obvious categories are "business stakeholders" and "QUASE-IOS users" (IT people). After having clarified the ontological definition of the notions of business stakeholder and QUASE-IOS customer (a part of the StakeQPA engineering studies) we are going to establish a more detailed categorization.

It is important to say here that in the framework of QUASE-IOS project we cooperate with two local IT companies focusing on custom software development for service-oriented and mobile systems: their developers will be the IT experts in the evaluations, whereas their customers will play the role of business stakeholders.

2. *Evaluation object:* the category of quality being evaluated, i.e. the quality of the prospective system under development vs. the quality of the QUASE-IOS software support and the ontological quality of the StakeQPA knowledge.

3. *Evaluation object source:* the source for the quality characteristics being evaluated, e.g. the appropriate software lifecycle process in case of external evaluation or the proposed quality forming process for the case of internal evaluation.

4. *Evaluation goal:* such goals differ significantly for external and internal evaluation cases.

5. *Evaluation method:* the empirical method used for the particular evaluation [6-9].

We can see the following major challenges of implementing the evaluation mechanisms for the QUASE-IOS project:

TABLE I. User evaluations in QUASE-IOS

Evaluation case	**Evaluation attributes**					
	Evaluation subjects	***Evaluation object***	***Evaluation object source***	***Evaluation time***	***Evaluation goal***	***Evaluation method***
internal *SIE*	Business stakeholders (roles specific for the SUD i.e. the problem at hand)	SUD quality attributes: performance, reliability etc.	Model-based PQ forming process: simulation, prototyping etc.	Runtime (SUD development lifecycle)	Forming the awareness data (e.g. NFR, design decision scores)	Model-based prototyping, quantitative data collection [7, 8]
external *SEE*	Business stakeholders (representative sample w.r.t. roles)	Interaction process quality attributes	Software process for forming the interaction support	Development time: EQ revealing process lifecycle	Interaction (EQ revealing) process forming and validation	Experiments [9], usability testing, case studies, non-pervasive techniques [7, 8]
external *CEE*	IT people: requirement engineers, project managers, analysts	Software solution quality attributes	QUASE-IOS solution lifecycle process	Development time: QUASE-IOS solution lifecycle	Solution validation	Surveys, interviews, prototyping, usability testing [7, 8]
external *OEE*	Business stakeholders, IT people: project managers, quality engineers etc.	StakeQPA ontology quality attributes	StakeQPA engineering process (according to e.g. [10])	Development time: ontology engineering lifecycle	StakeQPA forming and validation	Ontology evaluation methods [6]

1. External and internal evaluations have quite different goals and need to be implemented quite differently; on the other hand, some of them (*SIE* and *SEE*) need to be performed by the same category of users (the stakeholders without IT experience). We believe it is rather challenging to motivate such users to participate in these evaluations.

2. The differences between two categories of users involved in such evaluations, namely business stakeholders and the IT people, in particular, related to their perception of quality and the expectations of the proposed solution need to be addresses early and in full (e.g. on the ontology level).

3. The need of evaluating the interactions of the users of both categories with the same software solution involves establishing an interplay of the evaluations.

4. Embedding the empirical methods into the proposed software solution makes us face the performance problems as they need not hinder the interactive nature of this solution.

IV. CONCLUSIONS

As shown in this paper, the distinguishing features of the proposed approach are related to the fact that it requires organizing two categories of user evaluations – one being a part of the solution itself, one to evaluate the quality of the solution.

Such two-level structure of the evaluations makes their organization rather challenging (especially as the same category of users are supposed to perform the evaluations with quite different goals) and we would like to address these challenges by the discussion at the workshop.

REFERENCES

[1] R. P. L. Buse, C. Sadowski, and W. Weimer, "Benefits and barriers of user evaluation in software engineering research," SIGPLAN Not., vol. 46, pp. 643-656, 2011.

[2] V. A. Shekhovtsov, R. Kaschek, and S. Zlatkin, "Constructing POSE: a tool for eliciting quality requirements," in Information System Technology and Its Applications - ISTA 2007, LNI, vol. P-107. Bonn: GI, 2007, pp. 187–199.

[3] R. Kaschek, C. Kop, V. A. Shekhovtsov, and H. C. Mayr, "Towards simulation-based quality requirements elicitation: a position paper," in REFSQ 2008, LNCS, vol. 5025. London, Berlin, Heidelberg: Springer, 2008, pp. 135-140.

[4] A. Bachmann, W. Hesse, A. Ru, C. Kop, H. C. Mayr, and J. Vohringer, "A practical approach to ontology-based software engineering," in EMISA'07, LNI, vol. P-119. Bonn: GI, 2007, pp. 129-142.

[5] W. Hesse, "Ontologies in the software engineering process," in Proc. EAI'05, Ceur-WS.org, vol. 141, 2005.

[6] Z. Li, M. C. Yang, and K. Ramani, "A methodology for engineering ontology acquisition and validation," Artificial Intelligence for Engineering Design, Analysis and Manufacturing, vol. 23, pp. 37-51, 2009.

[7] J. Lazar, J. Feng, and H. Hochheiser, Research Methods in Human-Computer Interaction. Chichester: John Wiley & Sons, 2010.

[8] F. Shull, J. Singer, and D. I. K. Sjøberg, Eds., Guide to Advanced Empirical Software Engineering. London, Berlin, Heidelberg: Springer, 2008.

[9] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, Experimentation in Software Engineering: An Introduction. Boston, Dordrecht, London: Kluwer Academic Publishers, 2000.

[10] A. Gómez-Pérez, M. Fernández-López, and O. Corcho, Ontological Engineering. London, Berlin, Heidelberg: Springer, 2004.