2014 9th International Conference on the Quality of Information and Communications Technology

QuASE: A Tool Supported Approach to Facilitating Quality-Related Communication in Software Development

Vladimir A. Shekhovtsov, Heinrich C. Mayr, Vladyslav Lubenskyi
Application Engineering Group, Institute of Applied Informatics,
Alpen-Adria-Universität Klagenfurt, Klagenfurt, Austria
{volodymyr.shekhovtsov,heinrich.mayr,vladyslav.lubenskyi}@aau.at

***Abstract*— The paper describes the current state of the ongoing project QuASE. This project aims at facilitating quality-related communication in software development by the following means: a communication platform providing view harmonizing mechanisms for the different parties involved in a software process; decision making support related to quality issues; reuse of experience from former communication; prediction of quality-related experience. We describe usage scenarios for this approach, and outline its core concepts as well as the current progress of its implementation.**

***Keywords*- quality, understandability, decision support, software development**

I. Introduction

To succeed, software development processes require a smooth communication between the different parties involved, especially developers and business stakeholders. In particular, they need to have a common understanding of the quality (and its dimensions) the software under development should have; without this, quality-related problems are often detected only on later stages of the development lifecycle. Obviously, besides of enabling such communication, the communicated quality-related information has to be managed properly and made available during the software process; moreover, as past-experience may help to take the right decisions, such information should be provided in a way that allows for easy access and analysis.

To explore ways to a comprehensive solution for these issues, the QuASE project[1] has been established in cooperation with four small or medium sized software companies [9-12]. QuASE has two main goals:

(*goal G-1*) to establish means for enhancing the quality-related communication between the different parties involved in a software process, especially for developers and business stakeholders; these means are based on acquired and formalized domain knowledge about quality issues in software processes.

(*goal G-2*) to support decision making in the software process, reuse of quality-related experience, and the prediction of the future quality-related behavior of the involved parties – based on the collected knowledge.

This paper describes the current state of the QuASE project: In section 2 we define usage scenarios derived from the goals G-1 and G-2. Section 3 describes the current implementation status of the QuASE tool. Section 4 addresses related work, and is followed by conclusions and hints on future work directions.

II. QuASE Usage Scenarios

In the initial project stage, usage scenarios for the prospective tool have been elaborated in cooperation with the partner companies. These scenarios are presented subsequently broken down by four categories: *understandability management* (addressing G-1), *reuse* and *analysis* (both addressing G-2), and *preparation*. A more detailed treatment of understandability management may be found in [11].

A. Preparation

Prior to use the tool for a software development endeavor, the user has to specify the working environment in terms of context and document elements.

Following [10], we define *QuASE context elements* as units having particular views on communicated information, e.g. projects, organizations and their departments, involved stakeholders etc.. *QuASE documents* are defined as units shaping communicated information, e.g. JIRA [1] issues, requirement specifications etc..

For context selection the user chooses from the available set of context elements those which are relevant for the given project - thus forming *a context selector* (e.g. the particular stakeholders).

Document selection consists in picking the target document(s) out of the available set (e.g. a particular issue or a set of issues).

B. Understandability management

We distinguish understandability assessment and improvement scenarios.

1) Understandability assessment

Understandability assessment aims at identifying and assessing understandability problems: it involves calculating understandability metrics for the source document(s) according to the given context, and identifying potential understandability problems these document(s) might cause; also, recommendations for dealing with these problems are within that scenario.

For calculating the understandability metrics the user (1) selects the source context (the original context of the document; e.g., the originating company) and the target context (i.e., the context to check the document against; e.g., the customer representative), (2) selects the document or a set of documents, and (3) obtains the values characterizing

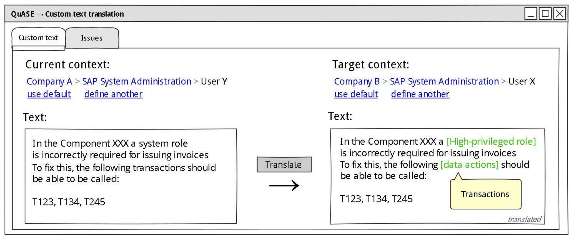
[1] The QuASE Project is sponsored by the Austrian Research Promotion Agency (FFG) in the framework of the Bridge 1 program (http://www.ffg.at/bridge1); Project ID: 3215531

the effort necessary to understand the document in the target context.

2) Understandability improvement

Understandability improvement is to minimize the (conceptual) distance between a given piece of communicated information and the perspective of the target party. QuASE aims at supporting translation-based and explanation-based improvement: the former focuses on purely terminological conflicts where simple substitution of terms is sufficient; the latter addresses problems resulting from one side's ignorance of the concepts behind the communicated terms.

For translation (see Fig.1) the user (1) selects the document or supplies the arbitrary text, (2) specifies the source and target contexts, and (3) obtains a version of the specified text with translated (quality or domain) terms.


QuASE → Custom text translation

Custom text | Issues

Current context:
Company A > SAP System Administration > User Y
use default define another

Text:
In the Component XXX a system role
is incorrectly required for issuing invoices
To fix this, the following transactions should
be able to be called:

T123, T134, T245

Translate

Target context:
Company B > SAP System Administration > User X
use default define another

Text:
In the Component XXX a [High-privileged role]
is incorrectly required for issuing invoices
To fix this, the following [data actions] should
be able to be called:

T123, T134, T245

Transactions

translated

Figure 1. Prototype UI for translation-based scenario

For explanation steps (1) and (2) are performed as above; in step (3) the user highlights the problematic terms in the document and obtains plain text or structured explanations of these terms.

C. Reuse

Reuse of information from past communication activities is based in our approach on applying similarity search techniques.

For *document-based similarity search* the user selects some attributes of a given document (e.g. a specific issue). The values of these attributes then are used for the search of documents in the systems' repository, the similarity distance of which is below a predefined threshold. *Example:* (1) the user selects the issue I and a subset of its attributes a_I including 'emotional tone' and 'expressed attitude to the project' and (2) gets the issues with similar values for these attributes.

For *context-based similarity search* the user selects a context element (e.g. stakeholder) with the goal of finding similar ones; the similarity distance is calculated based on a subset of attributes of the selected context element. In addition the system will return the documents that are related to the resulting elements.

Fig.2 shows an example of such scenario: the user (1) selects the stakeholder S and a subset of its attributes a_S including 'ability to explain' and 'attitude to detail'; (2) obtains the stakeholders with similar values for these attributes and the issues related to these stakeholders.

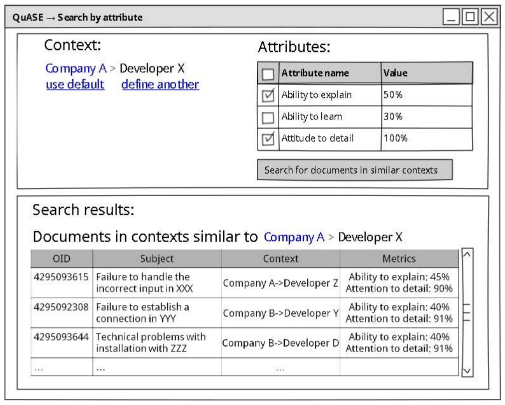
Similarity search techniques could also be used as building blocks for handling analysis scenarios.

D. Analysis

1) Recommendation

The user goal in such scenarios is to obtain recommendations, e.g. a document defining the way of working with a given issue, or for communicating with a particular stakeholder. We distinguish document-related and context-related recommendations, and classify these according to their scope: behavior-related recommendations describe the appropriate communication behavior; capability-related recommendations address user capabilities required to deal with the *recommendation target (RT)*, i.e., an issue, how to deal with recommendations are desired.

As an example, a user might select an issue I as RT, and obtains in reply the recommendation that 'a high level of IT knowledge' is required for dealing with I. In such cases, attribute values of the given document or context elements are used for an automatic recommendation extraction.


QuASE → Search by attribute

Context:
Company A > Developer X
use default define another

Attributes:

☐	Attribute name	Value
☑	Ability to explain	50%
☐	Ability to learn	30%
☑	Attitude to detail	100%

Search for documents in similar contexts

Search results:

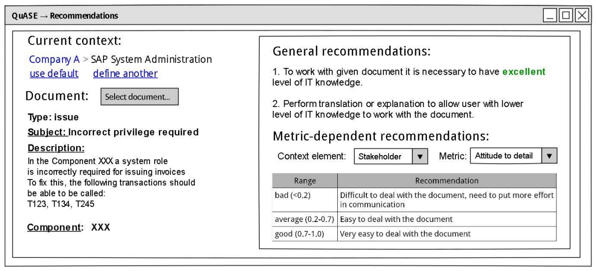
Documents in contexts similar to Company A > Developer X

OID	Subject	Context	Metrics
4295093615	Failure to handle the incorrect input in XXX	Company A->Developer Z	Ability to explain: 45% Attention to detail: 90%
4295092308	Failure to establish a connection in YYY	Company B->Developer Y	Ability to explain: 40% Attention to detail: 91%
4295093644	Technical problems with installation with ZZZ	Company B->Developer D	Ability to explain: 40% Attention to detail: 91%
...	...	...	

Figure 2. Prototype UI for context-based similarity search

In neither scenario the user selects a specific RT attribute and obtains in reply a set of recommendations related its value ranges. *Example:* the user selects an issue I and the attribute "Attitude to detail of the stakeholder"; this will result in recommendations for different value ranges of that attribute.

Fig.3 shows the prototype UI supporting both scenarios.


QuASE → Recommendations

Current context:
Company A > SAP System Administration
use default define another

Document: Select document...

Type: issue
Subject: Incorrect privilege required
Description:
In the Component XXX a system role
is incorrectly required for issuing invoices
To fix this, the following transactions should
be able to be called:
T123, T134, T245

Component: XXX

General recommendations:

1. To work with given document it is necessary to have **excellent** level of IT knowledge.

2. Perform translation or explanation to allow user with lower level of IT knowledge to work with the document.

Metric-dependent recommendations:

Context element: Stakeholder Metric: Attitude to detail

Range	Recommendation
bad (<0.2)	Difficult to deal with the document, need to put more effort in communication
average (0.2-0.7)	Easy to deal with the document
good (0.7-1.0)	Very easy to deal with the document

Figure 3. Prototype UI implementing recommendation scenarios

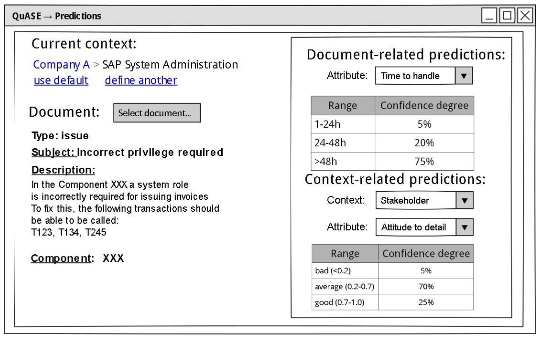
2) Prediction

Predictions are based on historical data from previous development projects available in the QuASE knowledge base. They support decision making, e.g. concerning communication strategies, or issue handling schedules (based on estimated processing times).

Computing predictions starts from attribute values of the given *prediction target* (PT), i.e., again documents or context elements.

In a *value prediction scenario* the user selects an attribute a_p of the PT p and obtains the predicted (computed) value of a_p. The computation is based on the value distribution for that attribute within the QuASE knowledge base restricted to elements similar to p. A more sophisticated prediction yields, instead of a single value, an ordered set of values (or value intervals), each ranked with a derived probability. Fig.4 shows the prototype UI for such scenario (with confidence degrees based on value range probabilities).

In a *value-based what-if analysis scenario* the user investigates the impact of changing metric values: in particular, the user can change the values for one (source) PT attribute (or a subset of attributes) and observe the changes of predicted values of another (target) attribute.


QuASE → Predictions

Current context:

Company A > SAP System Administration

use default   define another

Document: Select document...

Type: issue

Subject: Incorrect privilege required

Description:

In the Component XXX a system role
is incorrectly required for issuing invoices
To fix this, the following transactions should
be able to be called:
T123, T134, T245

Component: XXX

Document-related predictions:

Attribute: Time to handle

Range	Confidence degree
1-24h	5%
24-48h	20%
>48h	75%

Context-related predictions:

Context: Stakeholder

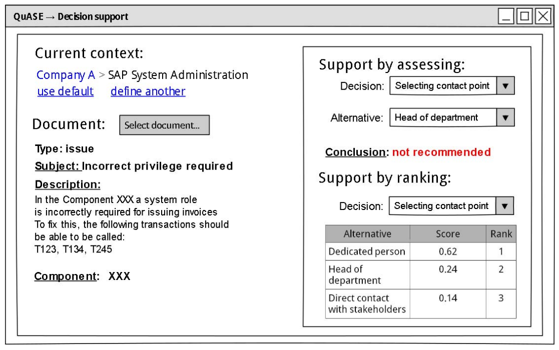
Attribute: Attitude to detail

Range	Confidence degree
bad (<0.2)	5%
average (0.2-0.7)	70%
good (0.7-1.0)	25%

Figure 4. Prototype UI implementing value prediction scenarios

3) Decision support

The goal of *decision recommendation* is to obtain, for a given decision case, recommendations based on prior decision data: either as assessments of the current decision alternatives or by ranking alternatives according to particular criteria. *Example:* the user selects an issue *I* and, as decision type "selecting contact point for communication"; as a result, he obtains a ranked set of alternatives for this decision extracted from prior decisions made for similar issues. Fig.5 shows the prototype UI supporting such scenario.


QuASE → Decision support

Current context:

Company A > SAP System Administration

use default   define another

Document: Select document...

Type: issue

Subject: Incorrect privilege required

Description:

In the Component XXX a system role
is incorrectly required for issuing invoices
To fix this, the following transactions should
be able to be called:
T123, T134, T245

Component: XXX

Support by assessing:

Decision: Selecting contact point

Alternative: Head of department

Conclusion: not recommended

Support by ranking:

Decision: Selecting contact point

Alternative	Score	Rank
Dedicated person	0.62	1
Head of department	0.24	2
Direct contact with stakeholders	0.14	3

Figure 5. Prototype UI for decision recommendation scenarios

The *decision-based what-if analysis scenario* describes situations, in which the user investigates the impact of making a particular decision: in detail, he selects alternatives and observes their effect on the value of relevant target attributes. E.g., the user might study, how selecting different contact points for dealing with issue *I* influences *I*'s time to handle.

III. IMPLEMENTATION ISSUES

A. Core Implementation Concepts

In this section, following [10], we present the set of core implementation concepts that form the foundations of the QuASE knowledge base (QuIRepository); the structure of this knowledge is depicted in Fig.6.

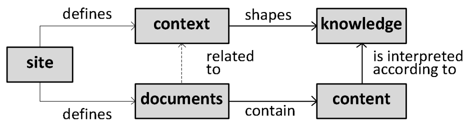


Figure 6. Generic structure of QuIRepository

1. *QuASE site:* owner of the given QuASE installation, e.g. a particular software provider; it defines the structure for context elements and documents.
2. *QuASE context* and *QuASE documents:* correspond to the definitions provided in Section 3: document units can be related to particular context elements.
3. *QuASE knowledge:* quality and domain knowledge that is subject of communication and harmonization. We organize it into knowledge modules representing particular views belonging to context elements;
4. *QuASE content:* the information that has to be communicated. It is shaped by document units and interpreted according to the respective knowledge. Dealing with the content is decoupled from dealing with their holder documents; i.e., we can think of this content as of a uniform stream of data.

To support understandability management, the QuASE knowledge modules are organized into a modular ontology (QuOntology). We thus provide a framework for translating between world views [9, 10, 13] by switching between configurations of context-specific modules that are used for the content stream interpretation [11]. In reuse and analysis scenarios, the routines for similarity search, decision support, and classification operate with the above structures through the QuIRepository interface.

B. Knowledge and Data Acquisition

The current context and document configuration depends on the specific QuASE site configuration. In addition to using a modular ontology at runtime to support understandability management as outlined in [11], we propose a static ontology-based approach to define the QuIRepository structure at deployment time. We concentrate here on context and document configuration; a similar approach is defined to specify the configuration of available QuOntology modules.

In this approach, we start with a context/document domain ontology, that is customized by specializations to site-specific ontology for a given application environment. At deployment time, this ontology is used to generate the site-specific database structure. To acquire the ontological knowledge, a notation for specifying context and document configurations is used, which will be supported by the metamodeling tool ADOxx (http://www.adoxx.org) to allow the responsible IT people to easily create and modify the desired configuration. The obtained ontological knowledge is then converted into OWL and used for database schema generation.

After acquiring the knowledge and generating the site-specific QuASE data store, it is necessary to transfer the data from the existing repositories (JIRA databases, file-based repositories etc.) into this store. The process is described in [10]; it involves applying the mapping specifications generated from the above ontological structures; these specifications take into account the current site-specific database structure and the specifics of the source repository.

C. Current State of Tool Support Implementation

The overall architecture of the QuASE tool consists of three layers: On the back-end there is a Java application (QuASE core) responsible for the logic and data/knowledge access. The client's web application is implemented as single page JavaScript application, hosted on the Django Framework (Python). Back-end parts of the solution communicate with each other by RabbitMQ.

Currently, a team of four software developers together with a team leader work at different components of the system. As an example, we outline here the implementation of the context selector functionality. Its goal is to support the selection scenario depicted in Section 3.

On the current stage of implementation, the context selector consists of two parts: the organizational unit as a component of the company's structure, and a specific user or a group of users. The corresponding implemented scenario consists of the following steps: (1) selecting the organizational unit; (2) selecting the user or user group; (3) obtaining the list of documents from this context. Following the static ontology-based approach, the QuASE data schema is defined at deployment time based, in part, on the context/document ontology expressed in OWL2. To support this, the physical data model for the QuASE storage contains the tables corresponding to meta-classes, specific instances of these meta-classes and instance attributes (holding scalar values or the links to other instances).

IV. Related Work

The approach discussed here belongs to the category of solutions that facilitate storing, reusing, adapting, and analyzing the development knowledge (experience management solutions [7, 8]) based on applying knowledge management techniques to software engineering [2, 3].

Among these, from the point of view of addressed software process qualities, we can distinguish solutions separately addressing decision support quality [4] and understandability of the particular categories of development-related artifacts [5, 6, 14]; the problems of reusability of communicated information are not addressed broadly. An additional common problem is that, as a rule, such solutions only consider the view of IT people.

The advantage of the QuASE approach is that it aims at an integrated solution that

(1) targets understandability and reusability of communicated information, as well as decision quality,
(2) handles understandability of generic fragments of communicated information (with emphasize on software quality) which are contained in documents,
(3) takes into account the views of all involved parties.

V. Conclusions and Future Work

In this paper, we outlined the main goals of the QuASE project and described its usage scenarios. The proposed approach facilitates effective communication in the software process by reducing the effort required from the parties to understand each other, especially while dealing with software quality, and making possible learning from past communication experience.

Next we will complete the QuASE tool by implementing all usage scenarios; this also involves acquiring real-world knowledge and data into QuOntology modules and QuIRepository structures. The β-Version of the tool will then be rolled out to the partner companies for a first proof of concept in practice.

References

[1] (08.05.2014). JIRA Issue Tracking System. Available: http://www.atlassian.com/software/jira

[2] A. Aurum, R. Jeffery, C. Wohlin, and M. Handzic, Eds., Managing Software Engineering Knowledge. Berlin-Heidelberg: Springer, 2003.

[3] F. O. Bjørnson and T. Dingsøyr, "Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used," Information and Software Technology, vol. 50, pp. 1055-1068, 2008.

[4] E. Damiani, F. Fulvio, S. Oltolina, and G. Ruffatti, "Improving software process accountability with Spago4Q," in 2nd Workshop ATGSE, Beijing-Dicembre, 2008.

[5] E. Kamsties, A. von Knethen, and R. Reussner, "A controlled experiment to evaluate how styles affect the understandability of requirements specifications," Information and Software Technology, vol. 45, pp. 955-965, 2003.

[6] H. A. Reijers and J. Mendling, "A study into the factors that influence the understandability of business process models," IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, vol. 41, pp. 449-462, 2011.

[7] K. Schneider, Experience and Knowledge Management in Software Engineering. Berlin-Heidelberg: Springer, 2009.

[8] N. Sharma, K. Singh, and D. Goyal, "Can Managing Knowledge and Experience Improve Software Process?-Insights from the literature," Research Cell: An International Journal of Engineering Sciences, vol. 4, pp. 324-333, 2011.

[9] V. Shekhovtsov, H. C. Mayr, and C. Kop, "Harmonizing the Quality View of Stakeholders," in Relating System Quality And Software Architecture, I. Mistrik, R. Bahsoon, et al., Eds.: Elsevier, 2014, in print.

[10] V. A. Shekhovtsov and H. C. Mayr, "Managing Quality Related Information in Software Development Processes," in CAiSE-Forum-DC 2014, CEUR Workshop Proceedings, vol. 1164: CEUR-WS.org, 2014, pp. 73-80.

[11] V. A. Shekhovtsov and H. C. Mayr, "Towards Managing Understandability of Quality-Related Information in Software Development Processes," in ICCSA 2014, Part V., LNCS, vol. 8583, B. Murgante, S. Misra, et al., Eds.: Springer, 2014, pp. 572-585.

[12] V. A. Shekhovtsov, H. C. Mayr, and C. Kop, "Acquiring Empirical Knowledge to Support Intelligent Analysis of Quality-Related Issues in Software Development," in QUATIC 2012, J. P. Faria, A. Silva, and R. J. Machado, Eds. New York: IEEE, 2012, pp. 153-156.

[13] V. A. Shekhovtsov, H. C. Mayr, and C. Kop, "Towards Conceptualizing Quality-Related Stakeholder Interactions in Software Development," in UNISCON 2012, LNBIP, vol. 137, H. C. Mayr, C. Kop, et al., Eds. Berlin-Heidelberg: Springer, 2013, pp. 73-86.

[14] M. Svahnberg, T. Gorschek, M. Eriksson, A. Borg, K. Sandahl, J. Borster, and A. Loconsole, "Perspectives on Requirements Understandability -- For Whom Does the Teacher's Bell Toll?," in Proc. REET '08, 2008, pp. 22-29.