V.Shekhovtsov, H.C.Mayr: Towards Intelligent Handling of Quality Related Issues in Software Development – A Project Report. In: Wuksch D., Peischl B., Kop C. (eds.): Ausgewählte Beiträge zur Anwenderkonferenz für Softwarequalität Test und Innovation - ASQT 2012, pp. 113-129. Österreichische Computer Gesellschaft, Wien (2013)

TOWARDS INTELLIGENT HANDLING OF QUALITY-RELATED ISSUES IN SOFTWARE DEVELOPMENT – A PROJECT REPORT[1)2)]

Vladimir A. Shekhovtsov, Heinrich C. Mayr [3)]

Abstract
Establishing reliable communication between all stakeholders involved in a software process is important for coming to terms and agreements on the quality of the prospective software. However, quality is perceived differently by different stakeholder groups, in particular, business people often refuse talking about quality before they experience the system. In addition, past experience related to handling quality-related issues in a software process (in a sense of issue management systems such as JIRA) is often not managed properly. To taskle this problem, we aim at a tool-supported framework implementing a four-stage issue handling process. It involves acquiring the raw information about quality-related issues from the different parties in a software process and converting this information into a set of knowledge structures reflecting their views; converting and integrating these structures into a "global view"; solving analytical tasks such as facilitating knowledge reuse based on the global view; and converting and externalizing the global view back into the form that reflects the views of the different parties. The proposed solution is expected to reduce the time and effort for establishing a communication basis while discussing software quality, thus cutting costs and strengthening the mutual trust of the parties.

1. Introduction

Software processes require the involvement of the affected business stakeholders throughout the software development lifecycle in order to be successful. A prerequisite for such involvement is establishing a communication basis ("communication channels") between the parties in the process. In particular, such a channel is needed for coming to terms and agreements on the quality of the software under development, in particular during the elicitation of and the agreement on quality-

[1] Funded by Österreichischen Forschungsförderungsgesellschaft mbH (FFG), Projekt 838531
[2] Project partners: Institute for Applied Informatics, Alpen-Adria-Universität Klagenfurt, ilogs mobile software GmbH, CICERO CONSULTING GmbH, Bernhard Lanner GmbH, trinitec IT Solutions & Consulting GmbH
[3] Institute for Applied Informatics, Alpen-Adria-Universität Klagenfurt, Austria
{Volodymyr.Shekhovtsov,Heinrich.Mayr}@aau.at

related requirements. Without this, quality defects are often detected and complained by the stakeholders only when the software is made available for acceptance testing.

Establishing such channels is still an open challenge requiring research on the following aspects:

1. The stakeholders think in different conceptualizations of the real world and use different terminologies, especially when dealing with the quality of the software under development: agreeing on a communication channel is usually time-consuming and often fails;
2. Business people tend to refuse talking about quality before they experience the implemented system or just restrict themselves with generic terms e.g. "the performance must be good";
3. Quality-related process approaches are hard to reuse, the prediction of the participants' behavior in future quality-related interactions (e.g. while dealing with prototypes or mock-ups) is vague.

The solution of the problem with establishing communication channels depends on the proper management of *quality-related issues* in software development (we use the term "issue" in the same sense as it is used in issue management systems (IMS) such as JIRA [57] and Mantis [71]: an event in a software process that needs to be handled and involves different parties). Unfortunately, the past experience of handling quality-related issues is often not properly managed:

1. If at all collected, the experience reflects the views of the developer side of the process – not those of the customer side.
2. The factors influencing the decisions of the parties in a software process (both business stakeholders and developers) are not properly understood and collected [25, 87, 115].

However, without a proper handling of this knowledge the business stakeholders' opinions and expectations on software quality tend to be neglected or at least handled incompletely: as a result, the understanding of the desired quality of the system becomes biased towards the view of the software developers – a problem known as "the inmates are running the asylum" [29]. This is supported by evidence of the practice of development: e.g. a study of the current practice of Carinthian software houses revealed that out of all the quality characteristics only the usability is addressed on the early stages of the software lifecycle. In particular, handling the performance issues is often postponed until the system is deployed. Clearly, that approach comes with negative consequences for all parties, since fixing the problems with unsatisfied stakeholder expectations on the late stages of the software process is difficult and expensive and may even lead to the failure of the whole project.

Within this paper, we propose to elaborate a tool-supported framework for properly handling quality-related issues in the software process. This framework should allow establishing effective quality-related communication channels between the process stakeholders. We present an overview of the ongoing QuASE project aimed at such a framework; it has been established in cooperation with four local software development companies.

The paper is structured as follows. In Section 2, we review the current research on this topic and formulate the goals of the project. Section 3 gives an overview of the QuASE process and describes its steps in detail; Section 4 is devoted to the proposed architecture and implementation specifics of the QuASE tool; it is followed by conclusions.

2. Related Work and Project Goals

2.1.Knowledge- and experience management solutions

Most relevant to our aims are solutions that facilitate storing, reusing, or analyzing the relevant development knowledge. Such solutions, in particular, apply the existing body of research on knowledge management to the field of software engineering [10, 18, 93]. The techniques include establishing knowledge bases to support software engineering tasks, in particular, architectural design [13].

More specific, so-called experience management solutions [97], are related to managing past software engineering experience. This research has been started by Basili with introducing the concept of software experience factory [15] – an enterprise-level software solution for capturing the relevant software engineering experience over time. In [51, 96] this concept has been enhanced to capture the knowledge related to applying software prototypes.

With respect to our aims, these solutions bear the following shortcomings:

1. They do not specifically address quality-related issues, especially from the point of view of collecting the quality requirements for the prospective system;
2. They collect the experience only as viewed from the developer side; the business stakeholder's view is often ignored. Consequently, they do not support establishing a communication basis for the process parties;
3. Establishing full-scale experience management solutions could be too cost intensive especially for small and medium scale software companies.

2.2.Quality-driven development support solutions

Several approaches aim at addressing quality to drive particular software engineering tasks or the entire software process, but are not based on knowledge or experience management techniques. Among these approaches are Quality-Driven Re-Engineering [112], approaches targeting quality-driven development restricted to the architectural design phase [61], and techniques that aim at making the entire software process driven by quality without organizing direct interaction with stakeholders [35, 45, 55, 75]. These solutions, however, lacking knowledge management or experience management functionality, do not directly address our aims: they do not allow for collecting, managing, reusing, and analyzing quality-related knowledge and experience throughout the software lifecycle or for establishing a quality-related communication basis for the parties in the software process.

2.3.Implementation-related problems

In addition to the shortcomings of the named solutions, we can distinguish common implementation-related problems: being developed as separate tools, these solutions are hard to integrate into the existing development process of the companies; in particular, they cannot access the data available in the existing development support systems such as issue management systems, project management tools, or wikis.

2.4.Project goals and expected results

To address these shortcomings, the QuASE project is aimed at defining theoretical foundations, elaborating implementation procedures and a tool support for

1. Acquiring and formalizing domain knowledge about handling quality-related issues in the software process;
2. Collecting the raw information about such issues from the involved parties and converting it into operational knowledge (using available domain knowledge to ensure conversion correctness);
3. Using the collected knowledge for establishing a quality-related communication basis for the involved parties, supporting decision making, reuse of quality-related experience, and the prediction of the future quality-related requirements of the involved parties.

We expect the project to bring the following primary results:

1. The theoretical foundations of a framework for collecting and sharing the operational knowledge regarding quality-related issues in a software process which supports the co-operative elicitation and analysis of quality-related requirements that allows the concerned parties for using their individual terminologies;
2. A set of formal approaches as the basis of practical implementation procedures for this framework;
3. The software tool (QuASE tool) implementing these practical procedures; this tool will have interfaces to share data with common issue management systems like JIRA, Mantis and others.

3. The QuASE Process

We plan to establish a four-stage *QuASE process* (Figure 1).

Figure 1. Stages of the QuASE process

1. The *elicitation stage* is devoted to acquiring the raw information about quality-related issues from the different parties in a software process and converting this information into a set of semantic knowledge structures reflecting the views of these parties;
2. The *integration stage* is devoted to converting and integrating (including conflict resolution) the results of stage 1 into a "global view";
3. During the *analysis stage* the analytical tasks such as facilitating knowledge reuse or predicting the handling of the future issues are solved based on the global view;
4. The *dissemination stage* is devoted to converting and externalizing the global view back into the form that reflects the views of the different parties.

3.1.QuOntology and QuIRepository

The process is based on managing the knowledge about quality-related issues in a repository that serves as an experience base (*QuIRepository*) and is structured along an ontology (*QuOntology*). QuOntology aims at sharing the conceptual knowledge about software quality, supplying the semantics to supplement the information about quality-related issues before converting it into the knowledge structures to be stored into QuIRepository; it serves as a foundation of the QuIRepository by defining its structure.

To establish *QuOntology*, we have already started empirical studies in cooperation with industry partners [101] coming up with the following categories of knowledge about quality-related aspects of interest:

1. The phenomenon of software quality, different types of such quality;
2. The categories of stakeholders participating in the interaction process, the ways of their selection;
3. The differences in perception of quality for different categories of stakeholders;
4. Quality-related stakeholder interaction process as it currently performed in industry (as seen by developers);
5. The properties of software projects influencing the ways of obtaining stakeholder opinions on software qualities;
6. The factors influencing stakeholder opinions on quality during the negotiation process (as seen by developers).

The obtained qualitative data is subject of open and selective coding and concept analysis activities as defined for grounded theory (we follow the approach of Corbin and Strauss [30] and take into account the specifics of applying this methodology to software engineering [3, 28]). As a result, a set of concepts to be included in QuOntology is being elaborated. This research has been already completed for the stakeholder interaction process and for the transformation of the quality concepts between its stages [102].

The theoretical work related to establishing QuOntology is related to defining a set of necessary conceptualizations on three levels:

1. On the *foundational level*, we rely on existing formal ontologies, possibly extended for our purposes. We choose to reuse concepts from the DOLCE ontology [73] with modifications related to the specific ways of representing quality-related knowledge [85]. As DOLCE relies in its representation of quality on the notion of conceptual space per Gärdenfors [44, 88] we plan to investigate the ways of applying this notion and its modified forms [86] for our purpose. In this, we follow our research on the conceptualization of quality [100].
2. The *domain level* is based on the conceptual foundations provided by the foundational level. It collects the notions specific for the given domain of handling quality-related issues involving business stakeholders. Some of these notions are supposed to be collected in empirical studies; in addition, at this level we plan to reuse the existing ontologies of e.g. people roles [19, 74], organizations [6, 20], and requirements [58]. We have already established such concepts for some subdomains, e.g. the concept of quality-related interaction process [102].
3. On the *application level*, following the Ontology-Based Software Engineering paradigm [49], QuOntology exposes itself through a set of application-level ontologies supporting the specific QuASE stages.

QuIRepository is intended for storing the operational knowledge about quality-related issues as instances of the QuOntology concepts. For the general principles of establishing the QuIRepository, the concept of experience knowledge base [15, 97] needs to be extended to enable collecting and sharing quality-related experience. QuIRepository has to be organized into subsections corresponding to the knowledge supporting the various stages of the QuASE approach by applying knowledge modularization techniques [109]. In this, we follow [12, 39], who propose a knowledge base that facilitates software engineering activities on different stages.

3.2. Elicitation stage

This stage is devoted to acquiring the raw information on quality-related issues from the different parties of the software process (business stakeholders, developers, project managers at both customer and the developer side etc.), supplementing it by rich semantics obtained from QuOntology, and storing the resulting knowledge into QuIRepository. We concentrate on collecting *party-dependent knowledge* based on the views and understanding of the particular party (i.e. its specific language) from the following sources:

1. The systems deployed at the developer side to support the software process (IMS, wikis etc.);
2. The process of *instrumented stakeholder interaction* where business stakeholders interact with QuASE-enhanced development support artifacts (e.g. prototypes and mock-ups) in the usual way, but the information about stakeholders' reactions to stimuli (e.g. mouse clicks or selected control paths) is transparently captured and collected into QuIRepository;
3. Direct stakeholder communication specifying the assessments of quality characteristics, such as usability assessments expressed while interacting with a mock-up or prototype.

Figure 2 illustrates an exemplary situation. Suppose that the stakeholders X (*project manager*) and Y (*developer*) state their knowledge on the issue A (*a particular performance issue*) in their terminology, whereas stakeholder Z (*business person on the customer side*) uses his terminology for stating his knowledge on the issue B (*a particular usability issue*). Neither Y knows the terminology of business stakeholder Z, nor does Z understand the terminology of the IT-persons X and Y. These problematic links are denoted by question marks in Figure 2.

X and Y provide their knowledge about issue A via an external IMS, Z provides his knowledge about issue B via instrumented interaction with a prototype. This raw data is supplemented with the semantics from QuOntology (such as the knowledge about the stakeholders or their roles, the type of the project etc.). The results are stored into the party-dependent knowledge section PD of QuIRepository, where each view has its separate subsection. I.e., knowledge about a particular issue can be found in different subsections corresponding to the respective views.

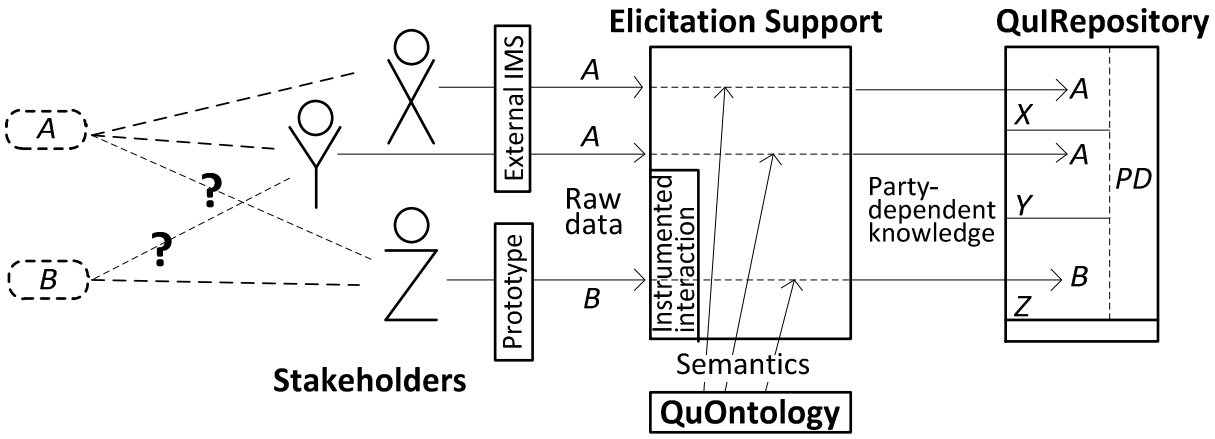

Figure 2. Elicitation stage knowledge conversions

To establish a theoretical basis for the acquisition of the raw information from the parties and converting it into knowledge, we plan to conceptualize the data to be collected and the process of

data collection. We base this conceptualization on the results of QuASE preliminary research [101, 102] together with the techniques of conceptual modeling of software process artifacts [78, 80, 125]. The theoretical concept of *quality-related issue* will be defined generalizing the artifacts available in existing IMS; in particular, we plan to treat software requirements and software negotiation scenarios as specific categories of issues [102, 125]. We propose to formalize the process of eliciting raw data by applying software process modeling techniques [78, 106]. To formalize the structure of the party-dependent knowledge, we apply the techniques of ontology-based knowledge acquisition [1, 8, 63, 124] and experience-knowledge transformation [94].

To establish a theoretical basis for the support for instrumented interaction, we plan to generalize the research results related to conceptualization of software prototypes [62, 66, 96] and quality simulation models [16, 31, 91] by introducing the concept of *interaction support solution*. The conceptualization of the process of stakeholder interaction is based on QuASE preliminary results [102] together with the techniques of conceptualization of the stakeholder negotiation [4].

3.3. Integration stage

This stage is devoted to establishing an integrated *party-independent* and *party-specific* representation of the collected issues that is suitable as a foundation for the communication basis. The party-independent knowledge is supposed to be integrated into a coherent whole and party-specific knowledge is supposed to be kept separate for every party. We plan to establish the classifier system which could be trained on past conversion samples, so the more data is fed into the system, the better it should be able to separate the knowledge. QuOntology can be used to facilitate this conversion.

The input for this stage consists of the results of the elicitation stage (party-dependent knowledge about quality-related issues). The result is supposed to be separated and integrated party-independent knowledge (stored in the PI section) and party-specific knowledge stored in the PS section of QuIRepository.

Figure 3 shows how the party-independent knowledge is separated from the party-specific knowledge for all the facts (e.g. the common knowledge about performance-related issue A is extracted out of the fact of the knowledge about A expressed by X). It is then integrated into the set of knowledge structures related to the particular issues (distributed into the subsections e.g. for the knowledge about A and B) and into the common knowledge relevant to all or some of the issues (marked by ∑). It is important to note that party-independent knowledge is free only from the views possessed by the particular parties (e.g. their own languages and world views), not from the facts about the particular parties; e.g. the fact that the project manager X is responsible for handling the performance-related issue A can be a part of such knowledge. The party-specific knowledge, after filtered out, is integrated into the knowledge structures related to the particular parties.

To establish the theoretical foundations for the separation of knowledge at this stage, we plan to apply the following methods and techniques:

1. *Sentiment analysis and opinion mining techniques* [68, 79] to separate opinion-based party-specific knowledge from the pure facts comprising party-independent knowledge [128].
2. *Knowledge classification and clustering techniques* [47]: classification methods (including learning classifier systems [34, 117]) correspond to the situation when the structure of the knowledge to be categorized is known [17, 119], whereas clustering [82, 113] can be applied to

discover this structure prior to categorizing the knowledge according to it [77]; we also plan to apply fuzzy clustering [53, 116] to account for the fuzziness of the knowledge;
3. *Formal concept analysis* [83, 110] to establish the rules for concept similarity in QuOntology and QuIRepository [32, 42];

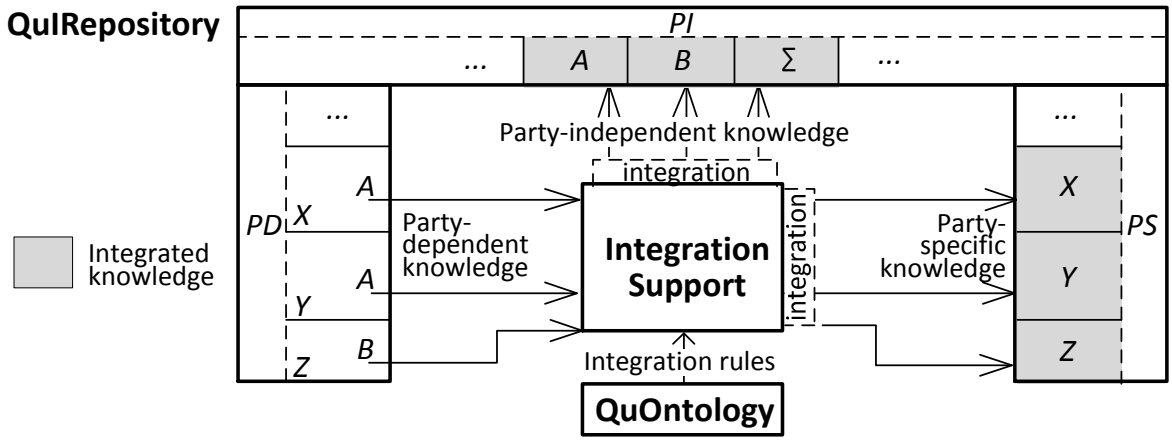

Figure 3. Integration stage knowledge conversions

Integrating the knowledge will be based on applying knowledge integration techniques such as ontology-based knowledge integration [36, 120, 121], multiple-criteria knowledge aggregation [127], ontology mediation [33, 41], and model merging [90].

3.4. Analysis stage

This stage processes integrated party-independent knowledge (free from the views of particular parties but containing all the relevant facts) following the user requests. The input data is obtained from the PI section of QuIRepository, the results are also stored there (in a separate subsection for analyzed knowledge). The supported activities are as follows:

1. Supporting decision making in the software process based on the knowledge about past quality-related issues;
2. Predicting the behavior of the parties during the stakeholder interaction process based on the knowledge on the past interactions (e.g. depending on the similarities between current issue and the historical data);
3. Finding the correlation between specific categories of stakeholders and the results of handling of the specific categories of issues or the specifics of interactions with other stakeholders.

Figure 4 illustrates the situation where the project manager X makes the analytical request Q, e.g. asking for some reusable piece of quality-related information. This request is processed using the *non-analyzed* party-independent knowledge; as a result, the *analyzed* party-independent knowledge is stored back into QuIRepository.

To establish theoretical foundations for the analysis stage, we apply the following methods:

1. *Decision making techniques* [14, 52, 92] (including multiple criteria [40] and fuzzy [22, 23, 70] ones) to support making analytic decisions in the software process, in particular artifact selection decisions [5, 67, 111];

2. *Case-based reasoning* [103, 114] and *concept similarity* [42, 122, 123] *techniques* to support reuse decisions;
3. *Prediction techniques* [104, 118], in particular, dealing with the behavior of the parties [65, 99, 107].

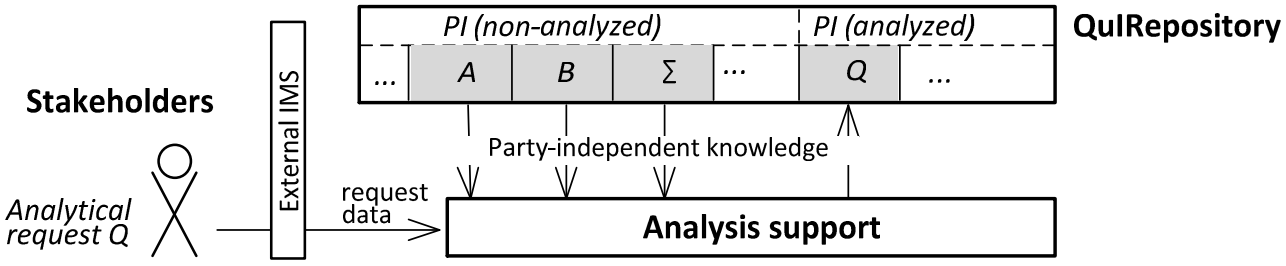

Figure 4. Analysis stage knowledge conversions

3.5.Dissemination stage

The *dissemination stage* is devoted to adapting the acquired knowledge and, in particular, the analysis results, to the terminologies and world views of the specified target parties. It uses party-specific knowledge of the target party to transform party-independent knowledge originated from both integration and analysis stages back into the party-dependent form. This conversion facilitates establishing a communication basis for the parties, when the language and the world view of the target party are applied to the generic knowledge about the issues under discussion originated from the other parties.

Figure 5 illustrates the situation where Z is provided with all existing information about issue A in his own language. Similarly, X receives the result of his analysis request in his language. All obtained party-dependent knowledge is stored into the PD section of QuIRepository forming a knowledge cache to be used in future processing.

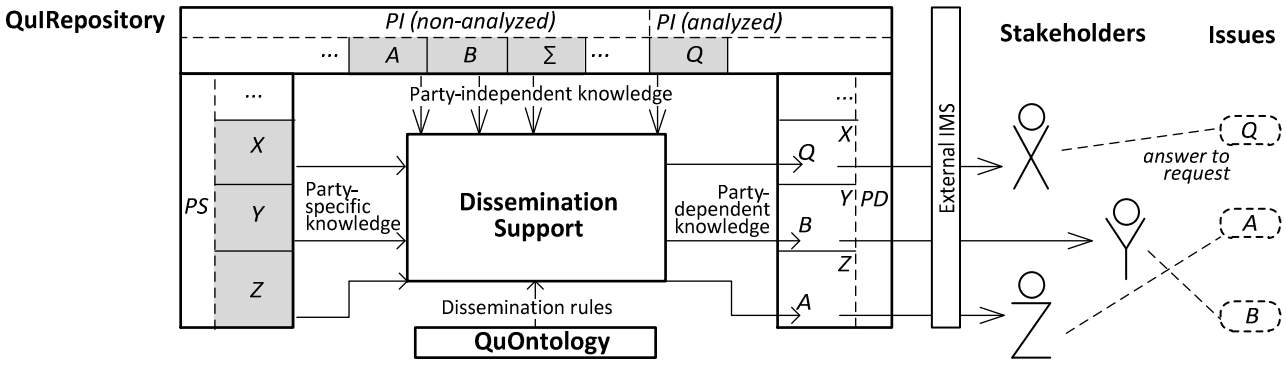

Figure 5. Dissemination stage knowledge conversions

The basic principles of establishing the theoretical foundations for this phase are similar to those established for the integration phase as they are supposed to rely on the same set of knowledge integration techniques. To integrate the ontology-based structure of party-specific and party-independent knowledge, we plan to apply ontology mediation techniques [33], in particular, ontology merging [24, 69, 89], they are supposed to form the base for ontology translation [37,

105]. We also apply the techniques of ontology-based knowledge representation [46], in particular, specific ontology representation patterns [50]. To formalize end-user representation of the relevant knowledge, we employ the techniques for conceptual modeling of user interfaces [59, 80, 81].

4. Implementation issues

We aim at the following goals while elaborating the practical implementation of the project:

1. The software solution needs to support all QuASE stages.
2. It needs to be possible to integrate the tool with existing development support systems (e.g. IMS) already deployed at the IT company or the customer site: in this case it should not force its users to learn new work paradigms.
3. It needs to seamlessly collect current issue data from the existing IMS, if available, ask for additional issue-related information using familiar interface; and to obtain the access to the past issue information.
4. It should be able to support different IMS available in industry, at least JIRA [57], Mantis [71], SAP Solution Manager [95], and XEOX [126] to allow the customer employing several IMS to collect data from all of them.
5. It should be extendable to allow for integration with IMS not originally considered.
6. It should also offer a standalone version not requiring external IMS (e.g. to be used by consulting companies).

To meet these requirements, we propose the architecture for the solution shown on Figure 6. To ensure that the target solution is IMS-independent, we aim at designing its architecture as containing a set of *IMS adapter modules* for existing IMS. For QuASE project, we propose to implement the adapters for JIRA, Mantis, SAP Solution Manager, and XEOX. These modules serve as facades for the rest of the solution. We also plan to have a module implementing standalone data collection functionality; it should also expose itself through an adapter interface.

Also, we plan to separate the set of modules implementing the main QuASE functionality from the separate *QuASE core* responsible only for module coordination and data exchange. QuASE core needs to provide an interface which should support creating the external adapters; we plan to use existing modularity support frameworks for the Java platform such as OSGi [48].

The modules comprising the QuASE architecture belong to the following categories:

1. *QuOntology and QuIRepository support modules* are responsible for providing access to these artifacts; they are planned to utilize Java ontology access features such as OWL APIs [54] or code mapping solutions [108].
2. *Stage-specific modules* (*elicitation, integration, analysis,* and *dissemination* modules), coordinated by QuASE core, are responsible for the implementation of the business logic related to the particular QuASE stages. The elicitation module should provide the features aiding in organizing instrumented interaction, in particular, an interaction solution registration interface.
3. *IMS Adapter modules* which need to conform to the two sets of external conventions (Figure 7): (1) plugin interface QJ provided by the QuASE core; (2) external plug-in interface JQ provided by the existing IMS, e.g. JIRA [56, 113] or Mantis [72] plugin interface.

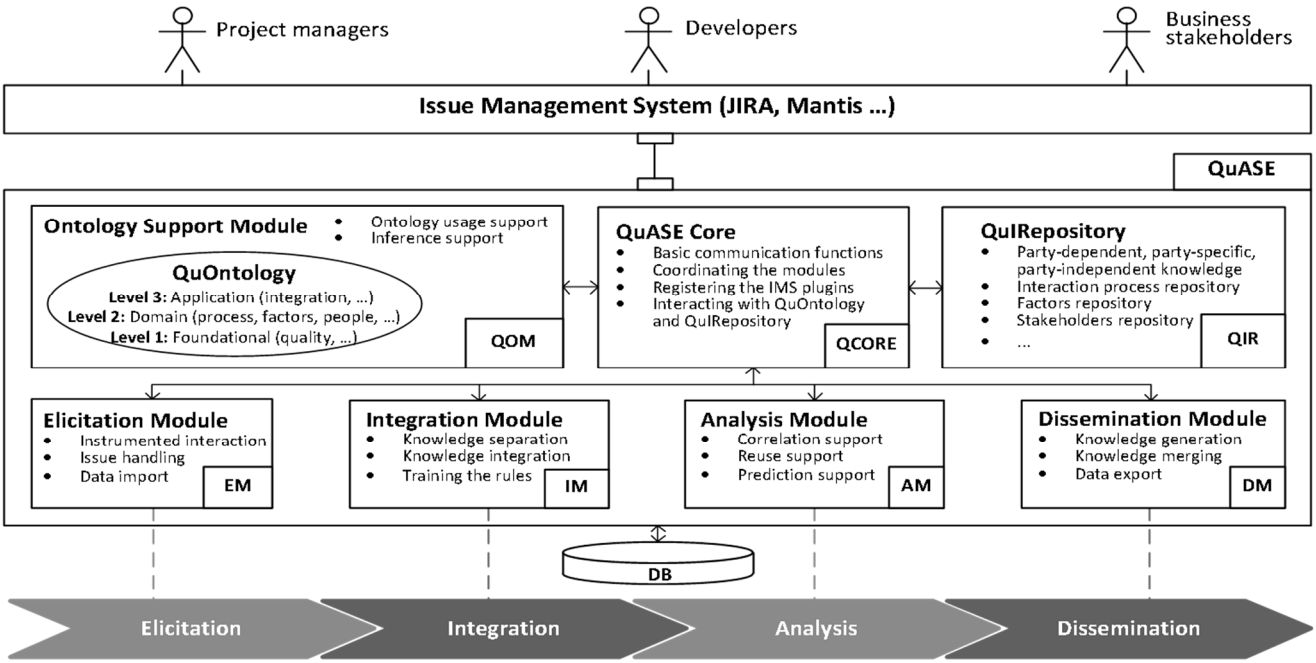

Figure 6. QuASE implementation architecture

The adapter modules should provide the following main functionality:

1. Forming the user interface in a way compatible with the particular IMS (e.g. as a set of Velocity [7] hypertext templates for JIRA); this interface needs to e.g. allow specifying the extended properties of the issue-related information according to the semantics provided by QuOntology;
2. Obtaining current and legacy issue data from the external IMS;
3. Performing two-way conversion between IMS-independent and IMS-dependent representation of the relevant information.

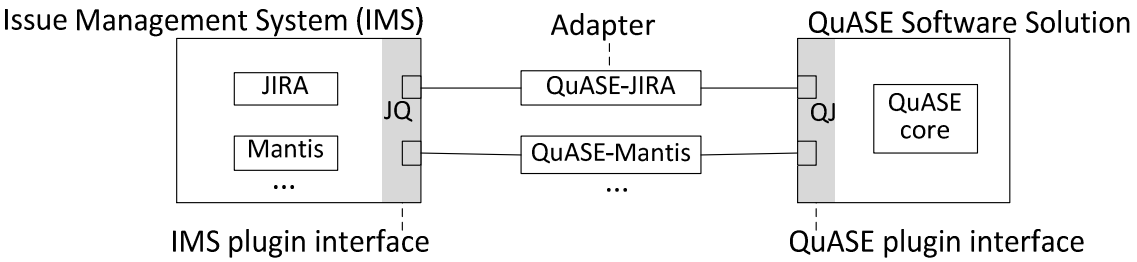

Figure 7. QuASE adapter modules

Similar functionality (but not relying on external IMS) needs also be implemented by the module responsible for supporting standalone execution. It should interactively collect the IMS-independent issue data directly from the end users through the standalone user interface. It also has to conform to the QJ interface.

5. Conclusions and future research directions

We have presented a currently ongoing project for handling quality-related issues in software development based on the experience management paradigm. We expect it to bring the following benefits:

1. Reducing time and effort for establishing a communication basis while talking about software quality for different parties in a software process, thus cutting related costs; this will also strengthen the mutual trust of the parties and allows for better understanding of the needs of business stakeholders;
2. Allowing for early and accurate capturing of quality-related knowledge originating from all involved parties: making this knowledge collected and available enriches the possibilities of the parties, allows them to make better-founded negotiation and development decisions;
3. Saving the effort of the developer team for establishing the quality-related communication channels by allowing the reuse of the past issue-handling experience and the prediction of the future quality-related behavior of the parties.

To our knowledge, there were no research attempts up to now to establish a communication basis for quality-related negotiations in the software process and to predict the behavior of the parties in such process based on the knowledge acquired from the information about quality-related stakeholder interaction and past quality-related issues.

Future investigations can be related to the following problem areas:

1. Extending the QuASE approach to handle quality-related issues in the development of software product lines [26, 84];
2. Making the QuASE approach seamlessly integrated into different agile software development processes [27, 64] such as Scrum [98] and XP [2] (e.g. by integrating with existing agile-supporting IMS solutions such as Greenhopper [9]);
3. Further investigating the application of the instrumented interaction paradigm, in particular, covering instrumented interactions based on the simulated usage processes [43, 60];
4. Integrating QuASE with industrial "live prototyping" solutions (such as iRise Studio [56] and Axure RP [11]; the survey is in [76]) which allow non-programmers to build and execute interactive software imitations to get stakeholder's feedback; it can be done by registering their prototypes as instrumented solutions;
5. Making QuASE cover the development of the process-aware information systems [38] where the system under development is based on the particular representation of the business process (e.g. using BPMN [21] or other modeling notation) to be run by the specific process engine.

References

[1] ABECKER, A., ELST, L.: Ontologies for Knowledge Management. in: Staab, S., Studer, R. (eds.): Handbook on Ontologies. Springer, 2009, pp. 713-734

[2] ABRAHAMSSON, P., BASKERVILLE, R., CONBOY, K., FITZGERALD, B., MORGAN, L., WANG, X. (eds.): Agile Processes in Software Engineering and Extreme Programming. Springer, Berlin-Heidelberg, 2008

[3] ADOLPH, S., HALL, W., KRUCHTEN, P.: Using grounded theory to study the experience of software development. Empir Software Eng 16, 2011, pp. 487–513

[4] ADOLPH, S., KRUCHTEN, P., HALL, W.: Reconciling perspectives: A grounded theory of how people manage the process of software development. The Journal of Systems and Software 85, 2012, pp. 1269-1286

[5] AL-NAEEM, T., GORTON, I., BABAR, M.A., RABHI, F.A., BENATALLAH, B.: A quality-driven systematic approach for architecting distributed software applications. in: Roman, G.-C., Griswold, W.G., Nuseibeh, B. (eds.): Proc. 27th International Conference on Software Engineering (ICSE 2005). ACM, 2006, pp. 244-253

[6] ALMEIDA, J.P.A., CARDOSO, E.C.S.: On the Elements of an Enterprise: Towards an Ontology-Based Account. in: SAC'11. IEEE, 2011, pp. 323-330
[7] The Apache Velocity Project, http://velocity.apache.org/, accessed 22.08.2012
[8] AROYO, L., DENAUX, R., DIMITROVA, V., PYE, M.: Interactive ontology-based user knowledge acquisition: A case study. in: Proc. ESCW'06. 2006
[9] Atlassian Greenhopper, http://www.atlassian.com/software/greenhopper/, accessed 23.08.2012
[10] AURUM, A., JEFFERY, R., WOHLIN, C., HANDZIC, M. (eds.): Managing Software Engineering Knowledge. Springer, 2003
[11] Axure RP, http://www.axure.com/, accessed 23.08.2012
[12] BABAR, M.A.: Supporting the Software Architecture Process with Knowledge Management. in: Babar, M.A., Dingsøyr, T., Lago, P., van Vliet, H. (eds.): Software Architecture Knowledge Management. Springer, 2009, pp. 69-86
[13] BABAR, M.A., DINGSØYR, T., LAGO, P., VAN VLIET, H. (eds.): Software Architecture Knowledge Management: Theory and Practice. Springer, 2009
[14] BABAR, M.A., ZHU, L., JEFFERY, R.: A Framework for Classifying and Comparing Software Architecture Evaluation Methods. in: Proc. ASWEC'04. IEEE, 2004, pp. 309-318
[15] BASILI, V., CALDIERA, G., ROMBACH, D.H.: Experience factory. in: Marciniak, J.J. (ed.): Encyclopedia of Software Engineering. John Wiley & Sons, New York, 1994, pp. 469–476
[16] BENJAMIN, P., PATKI, M., MAYER, R.: Using ontologies for simulation modeling. in: WSC'06. 2006, pp. 1151-1159
[17] BJELLAND, T.K.: Classification: Assumptions and Implications for Conceptual Modeling, PhD Dissertation. Univ. of Bergen, 2004
[18] BJØRNSON, F.O., DINGSØYR, T.: Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used. Information and Software Technology 2008,
[19] BOELLA, G., VAN DER TORRE, L.: A Foundational Ontology of Organizations and Roles. in: Declarative Agent Languages and Technologies IV. Springer, 2006, pp. 78-88
[20] BOTTAZZI, E., FERRARIO, R.: A Path to an Ontology of Organizations. in: Guizzardi, G., Wagner, G. (eds.): VORTE'05. 2005, pp. 9-16
[21] Business Process Model and Notation (BPMN) Version 1.2. OMG, 2009
[22] BÜYÜKÖZKAN, G., FEYZIOĞLU, O.: Group decision making to better respond customer needs in software development. Computers & Industrial Engineering 48, 2005, pp. 427-441
[23] CARLSSON, C., FULLER, R.: Fuzzy Reasoning in Decision Making and Optimization. Springer, 2002
[24] CHEN, R.-C., BAU, C.-T., YEH, C.-J.: Merging domain ontologies based on the WordNet system and Fuzzy Formal Concept Analysis techniques. Applied Soft Computing 11, 2011, pp. 1908-1923
[25] CLARKE, P., O'CONNOR, R.V.: The situational factors that affect the software development process: Towards a comprehensive reference framework. Journal of Information Software and Technology 54, 2012, pp. 433-447
[26] CLEMENTS, P.C., NORTHROP, L.M.: Software product lines: practices and patterns. Addison-Wesley, 2001
[27] COCKBURN, A.: Agile Software Development. Addison-Wesley, Reading, MA, 2001
[28] COLEMAN, G., O'CONNOR, R.: Investigating software process in practice: A grounded theory perspective. The Journal of Systems and Software 81, 2008, pp. 772–784
[29] COOPER, A.: The Inmates are Running the Asylum. Sams, 2004
[30] CORBIN, J., STRAUSS, A.: Basics of Qualitative Research, 3rd ed. Sage Pubs., Thousand Oaks, CA, 2008
[31] CORTELLESSA, V., PIERINI, P., SPALAZZESE, R., VIANALE, A.: MOSES: MOdeling Software and platform architEcture in UML 2 for Simulation-based performance analysis. in: QoSA 2008. LNCS, Vol. 5281. Springer, 2008, pp. 86-102
[32] CROSS, V.V., WENTING, Y.: Formal concept analysis for ontologies and their annotation files. in: Proc. FUZZ-IEEE 2008. 2008, pp. 2014-2021
[33] DE BRUIJN, J., EHRIG, M., FEIER, C., MARTÍNS-RECUERDA, F., SCHARFFE, F., WEITEN, M.: Ontology Mediation, Merging, and Aligning. in: Semantic Web Technologies. Wiley, 2006, pp. 95-113
[34] DE JONG, K.A., SPEARS, W.M., GORDON, D.F.: Using Genetic Algorithms for Concept Learning. Machine Learning 13, 1993, pp. 161-188
[35] DE MIGUEL, M.A., MASSONET, P., SILVA, J.P., BRIONES, J.: Model Based Development of Quality-Aware Software Services. in: ISORC'08. IEEE, 2008, pp. 563-569
[36] DOERR, M., HUNTER, J., LAGOZE, C.: Towards a Core Ontology for Information Integration. Journal of Digital Information 4, 2003,
[37] DOU, D., MCDERMOTT, D., QI, P.: Ontology Translation on the Semantic Web. in: Journal on Data Semantics II. LNCS, Vol. 3360. Springer, 2005, pp. 35-57
[38] DUMAS, M., VAN DER AALST, W., TER HOFSTEDE, A. (eds.): Process-Aware Information Systems. Wiley-IEEE, 2005

[39] FELDMANN, R.L., GEPPERT, B., RÖßLER, F.: An Integrating Approach for Developing Distributed Software Systems – Combining Formal Methods, Software Reuse, and the Experience Base. in: Proc. ICECCS'99. IEEE, 1999, pp. 54-63
[40] FIGUEIRA, J., GRECO, S., EHRGOTT, M. (eds.): Multiple Criteria Decision Analysis:State of the Art Surveys. Springer, Heidelberg-Berlin, 2004
[41] FIRAT, A.: Information integration using contextual knowledge and ontology merging, PhD Thesis. Massachusetts Institute of Technology, 2003
[42] FORMICA, A.: Ontology-based concept similarity in Formal Concept Analysis. Information Sciences 176, 2006, pp. 2624-2641
[43] FRITZSCHE, M., PICHT, M., GILANI, W., SPENCE, I., BROWN, J., KILPATRICK, P.: Extending BPM environments of your choice with performance related decision support. in: BPM 2009. LNCS, Vol. 5701, 2009, pp. 97-112
[44] GÄRDENFORS, P.: Conceptual Spaces: A Geometry of Thought. MIT Press, Cambridge, MA, 2000
[45] GRAMBOW, G., OBERHAUSER, R., REICHERT, M.: Contextual Injection of Quality Measures into Software Engineering Processes. Int'l Journal on Advances in Software 4, 2011, pp. 76-99
[46] GRIMM, S., HITZLER, P., ABECKER, A.: Knowledge Representation and Ontologies. in: Studer, R., Grimm, S., Abecker, A. (eds.): Semantic Web Services: Concepts, Technology and Applications. Springer, 2007, pp. 51-106
[47] HALGAMUGE, S.K., WANG, L. (eds.): Classification and Clustering for Knowledge Discovery. Springer, 2005
[48] HALL, R.S., PAULS, K., MCCULLOCH, S., SAVAGE, D.: OSGi in Action. Manning Publications, 2011
[49] HESSE, W.: Ontologies in the Software Engineering process. in: Proc. EAI'05. Ceur-WS.org, Vol. 141, 2005
[50] HOEKSTRA, R.J.: Ontology Representation: design patterns and ontologies that make sense, PhD Thesis, Univ. of Amsterdam, 2009
[51] HOLMLID, S., EVENSON, S.: Prototyping and enacting services: Lessons learned from human-centered methods. in: Proc. 10th Quality in Services Conf., QUIS 10. Orlando, Florida. 2007
[52] HOOVER, C.L., ROSSO-LLOPART, M., TARAN, G.: Evaluating project decisions: case studies in software engineering. Addison-Wesley, 2010
[53] HÖPPNER, F., KLAWONN, F., KRUSE, R., RUNKLER, T.: Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition. Wiley, 1999
[54] HORRIDGE, M., BECHHOFER, S.: The OWL API: A Java API for Working with OWL 2 Ontologies. in: Proc. OWLED'09. CEUR Workshop Proceedings, Vol. 529, 2009
[55] HUMMEL, O., MOMM, C., HICKL, S.: Towards quality-aware development and evolution of enterprise information systems. in: Proceedings of the 2010 ACM Symposium on Applied Computing. ACM, 2010, pp. 137-144
[56] iRise Studio, http://www.irise.com/products/irise_studio, accessed 23.08.2012
[57] JIRA Issue Tracking System, http://www.atlassian.com/software/jira, accessed 28.05.2012
[58] JURETA, I., MYLOPOULOS, J., FAULKNER, S.: A core ontology for requirements. Applied Ontology 4, 2009, pp. 169-244
[59] KAINDL, H., CONSTANTINE, L., PASTOR, O., SUTCLIFFE, A., ZOWGHI, D.: How to Combine Requirements Engineering and Interaction Design? in: Proc RE'2008. IEEE CS Press, 2008, pp. 299-301
[60] KASCHEK, R., KOP, C., SHEKHOVTSOV, V.A., MAYR, H.C.: Towards Simulation-Based Quality Requirements Elicitation: A Position Paper. in: REFSQ 2008. LNCS, Vol. 5025. Springer, 2008, pp. 135-140
[61] KIM, S., KIM, D.-K., PARK, S.: Tool support for quality-driven development of software architectures. in: ASE'10. ACM, 2010, pp. 127-130
[62] LAWSON, J.-Y.L., AL-AKKAD, A.-A., VANDERDONCKT, J., MACQ, B.: An Open Source Workbench for Prototyping Multimodal Interactions Based on Off-The-Shelf Heterogeneous Components. in: EICS'09. ACM, 2009
[63] LEBBINK, H.-J., WITTEMAN, C.L.M., MEYER, J.-J.C.: Ontology-Based Knowledge Acquisition for Knowledge Systems. in: Blockdeel, H., Denecker, M. (eds.): Proc. BNAIC'02. 2002, pp. 195-202
[64] LEFFINGWELL, D.: Agile software requirements. Addison-Wesley, Upper Saddle River, NJ, 2011
[65] LEI, S., YIMING, Z., CHAO, X., XIA, H., BIYUN, H.: Predicting User Behavior in E-commerce Based on Psychology Model. Proc. FSKD'09, pp. 576-580. IEEE, 2009
[66] LIM, Y.-K., STOLTERMAN, E., TENENBERG, J.: The Anatomy of Prototypes: Prototypes as Filters, Prototypes as Manifestations of Design Ideas. ACM Transactions on Computer-Human Interaction 15, 2008, pp. Article 7
[67] LIN, H.-Y., HSU, P.-Y., SHEEN, G.-J.: A fuzzy-based decision-making procedure for data warehouse system selection. Expert Systems with Applications 32, 2007, pp. 939-953
[68] LIU, B.: Sentiment Analysis and Subjectivity. in: Indurkhya, N., Damerau, F.J. (eds.): Handbook of Natural Language Processing, Second Edition. Chapman and Hall/CRC, 2010, pp. 627-666
[69] LIU, S.-P., LI, G.-Y.: Formal concept analysis based ontology merging method. Comp. Eng. Design 32, 2011, pp. 1434-1437

[70] LU, J., ZHANG, G., RUAN, D., WU, F.: Multi-Objective Group Decision Making: Methods, Software and Applications With Fuzzy Set Techniques. Imperial College Press, 2007
[71] Mantis Bug Tracker, http://www.mantisbt.org, accessed 28.05.2012
[72] Mantis Bug Tracker Developers Guide, http://www.mantisbt.org/docs/master/en/developers.html, accessed 22.08.2012
[73] MASOLO, C., BORGO, S., GANGEMI, A., GUARINO, N., OLTRAMARI, A.: The WonderWeb Library of Foundational Ontologies. WonderWeb Deliverable D18. Ontology Library (final). . ISTC-CNR, Trento, 2003
[74] MASOLO, C., VIEU, L., BOTTAZZI, E., CATENACCI, C., FERRARIO, R., GANGEMI, A., GUARINO, N.: Social Roles and their Descriptions. in: Dubois, D., Welty, C.A., Williams, M.-A. (eds.): KR'2004. AAAI Press, 2004, pp. 267-277
[75] MATINLASSI, M.: Quality-Driven Software Architecture Model Transformation. in: Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture. IEEE Computer Society, 2005, pp. 199-200
[76] MEMMEL, T., GUNDELSWEILER, F., REITERER, H.: Prototyping Corporate User Interfaces - Towards a Visual Specification of Interactive Systems. in: Proc. IASTED-HCI 2007. 2007
[77] MINEAU, G., GODIN, G.R.: Automatic structuring of knowledge bases by conceptual clustering. IEEE Trans. Knowl. Data Eng 7, 1995, pp. 824–828
[78] MÜNCH, J., ARMBRUST, O., KOWALCZYK, M., SOTO, M.: Software Process Definition and Management. Springer, 2012
[79] PANG, B., LEE, L.: Opinion Mining and Sentiment Analysis. Found. Trends Inf. Retr. 2, 2008, pp. 1-135
[80] PASTOR, O., MOLINA, J.C.: Model-Driven Architecture in Practice. Springer, Berlin-Heidelberg, 2007
[81] PATERNÒ, F.: Model-based Design and Evaluation of Interactive Applications. Springer, 2000
[82] PEDRYCZ, W.: Knowledge-Based Clustering: From Data to Information Granules. Wiley, 2005
[83] POELMANS, J., ELZINGA, P., VIAENE, S., DEDENE, G.: Formal Concept Analysis in Knowledge Discovery: A Survey. in: Croitoru, M., Ferré, S., Lukose, D. (eds.): Conceptual Structures: From Information to Intelligence. LNCS, Vol. 6208. Springer, 2010, pp. 139-153
[84] POHL, K., BÖCKLE, G., VAN DER LINDEN, F.: Software Product Line Engineering: Foundations, Principles, and Techniques. Springer, 2005
[85] PROBST, F.: Ontological Analysis of Observations and Measurements. in: GIScience'06. 2006, pp. 304-320
[86] PROBST, F.: Observations, Measurements and Semantic Reference Spaces. Applied Ontology 3, 2008, pp. 63-89
[87] PROYNOVA, R., PAECH, B., WICHT, A., WETTER, T.: Use of Personal Values in Requirements Engineering – A Research Preview. in: Wieringa, R., Persson, A. (eds.): REFSQ 2010. LNCS, Vol. 6182. Springer, 2010, pp. 17-22
[88] RAUBAL, M.: Formalizing Conceptual Spaces. in: Proc. FOIS 2004. IOS Press, 2004, pp. 153-164
[89] RAUNICH, S., RAHM, E.: ATOM: Automatic target-driven ontology merging. Data Engineering (ICDE), 2011 IEEE 27th International Conference on, pp. 1276-1279, 2011
[90] RICHARDS, D.: Merging individual conceptual models of requirements. Requirements Engineering 8, 2003, pp. 195-205
[91] ROBINSON, S.: Conceptual modelling for simulation. J Operational Research Society 59, 2008, pp. 278-290
[92] RUHE, G.: Software Engineering Decision Support – A New Paradigm for Learning Software Organizations. in: Henninger, S., Maurer, F. (eds.): Advances in Learning Software Organizations. LNCS, Vol. 2640. Springer, 2003, pp. 104-113
[93] RUS, I., LINDVALL, M.: Knowledge Management in Software Engineering. IEEE Software 3, 2002, pp. 26-38
[94] SANIN, C., SZCZERBICKI, E., TORO, C.: An OWL Ontology of Set of Experience Knowledge Structure. Journal of Universal Computer Science 13, 2007, pp. 209-223
[95] SCHÄFER, M., MELICH, M.: SAP Solution Manager (3rd Edition). SAP Press, 2011
[96] SCHNEIDER, K.: Prototypes as Assets, not Toys: Why and How to Extract Knowledge from Prototypes (Experience Report). in: Proc. ICSE'96. IEEE, 1996
[97] SCHNEIDER, K.: Experience and Knowledge Management in Software Engineering. Springer, 2009
[98] SCHWABER, K., BEEDLE, M.: Agile Software Development with Scrum. Prentice Hall PTR,, 2001
[99] ŞERBAN, G., TARŢA, A., MOLDOVAN, G.: A Learning Interface Agent for User Behavior Prediction. in: Jacko, J. (ed.): Proc. HCI'07. LNCS, Vol. 4552. Springer, 2007, pp. 508-517
[100] SHEKHOVTSOV, V.A.: On the evolution of quality conceptualization techniques. in: Kaschek, R., Delcambre, L. (eds.): The Evolution of Conceptual Modeling. LNCS, Vol. 6520. Springer, 2011, pp. 117–136
[101] SHEKHOVTSOV, V.A., MAYR, H.C., KOP, C.: Acquiring Empirical Knowledge to Support Intelligent Analysis of Quality-Related Issues in Software Development. in: Faria, J.P., Silva, A., Machado, R.J. (eds.): Proc. QUATIC 2012. IEEE, 2012, pp. 153-156
[102] SHEKHOVTSOV, V.A., MAYR, H.C., KOP, C.: Towards Conceptualizing Quality-Related Stakeholder Interactions in Software Development. in: Proc. UNISCON 2012. LNBIP, Springer, 2013, in press

[103] SHEPPERD, M.: Case-Based Reasoning and Software Engineering. in: Aurum, A., Jeffery, R., Wohlin, C., Handzic, M. (eds.): Managing Software Engineering Knowledge. Springer, 2003, pp. 181-198
[104] SHEPPERD, M., CARTWRIGHT, M., KADODA, G.: On Building Prediction Systems for Software Engineers. Empirical Software Engineering 5, 2000, pp. 175-182
[105] SILVA PARREIRAS, F., STAAB, S., SCHENK, S., WINTER, A.: Model Driven Specification of Ontology Translations. in: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.): Conceptual Modeling - ER 2008. LNCS, Vol. 5231. Springer, 2008, pp. 484-497
[106] Software Process Engineering Metamodel (SPEM) 2.0. OMG, 2008
[107] SONG, J., TANG, E., LIU, L.: User Behavior Pattern Analysis and Prediction Based on Mobile Phone Sensors. in: Ding, C., Shao, Z., Zheng, R. (eds.): Network and Parallel Computing. LNCS, Vol. 6289. Springer, 2010, pp. 177-189
[108] STEVENSON, G., DOBSON, S.: Sapphire: Generating Java Runtime Artefacts from OWL Ontologies. in: Salinesi, C., Pastor, O. (eds.): Proc. CAiSE 2011 Workshops. LNBIP, Vol. 83. Springer, 2011, pp. 425-436
[109] STUCKENSCHMIDT, H., PARENT, C., SPACCAPIETRA, S. (eds.): Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization. Springer, 2009
[110] STUMME, G.: Formal Concept Analysis. in: Staab, S., Studer, R. (eds.): Handbook on Ontologies. Springer, 2009, pp. 177-199
[111] SVAHNBERG, M., WOHLIN, C., LUNDBERG, L., MATTSSON, M.: A Quality-Driven Decision-Support Method for Identifying Software Architecture Candidates. Intel J Software Eng and Knowledge Eng 13, 2003, pp. 547-573
[112] TAHVILDARI, L., KONTOGIANNIS, K., MYLOPOULOS, J.: Quality-Driven Software Re-engineering. J. of Systems and Software 66, 2003, pp. 225–239
[113] TANG, Y., SYCARA, K., SENSOY, M., PAN, J.Z.: Survey on clustering methods for ontological knowledge. International Technology Alliance, 2012
[114] TAUTZ, C., ALTHOFF, K.-D., NICK, M.: A case-based reasoning approach for managing qualitative experience. in: Proc. AAAI'00. 2000
[115] THEW, S., SUTCLIFFE, A.: Investigating the Role of 'Soft issues' in the RE Process. in: Proc. RE'2008. IEEE, 2008, pp. 63-66
[116] TRAPPEY, A.J.C., TRAPPEY, C.V., HSU, F.-C., HSIAO, D.W.: A Fuzzy Ontological Knowledge Document Clustering Methodology. IEEE Transactions on Systems, Man, Cybernetics 39, 2009, pp. 123–131
[117] URBANOWICZ, R.J., MOORE, J.H.: Learning classifier systems: a complete introduction, review, and roadmap. J. Artif. Evol. App. 2009, 2009, pp. 1-25
[118] VANDECRUYS, O., MARTENS, D., BAESENS, B., MUES, C., DE BACKER, M., HAESEN, R.: Mining software repositories for comprehensible software fault prediction models. Journal of Systems and Software 81, 2008, pp. 823-839
[119] VELOSO, A., MEIRA, W.: Demand-Driven Associative Classification. Springer, 2011
[120] VISSER, U.: Intelligent Information Integration for the Semantic Web. Springer, 2004
[121] WACHE, H., VOEGELE, T., VISSER, U., STUCKENSCHMIDT, H., SCHUSTER, G., NEUMANN, H., HUEBNER, S.: Ontology-Based Integration of Information - A Survey of Existing Approaches. in: Proc. IJCAI'01. 2001, pp. 108-118
[122] WANG, L., GONG, D.: A structural information method for evaluating concept similarity. in: Proc. FSKD'10. 2010, pp. 1966-1970
[123] WANG, L., LIU, X.: A new model of evaluating concept similarity. Knowledge-Based Systems 21, 2008, pp. 842-846
[124] WANG, X., CHAN, C.W., HAMILTON, H.J.: Design of knowledge-based systems with the ontology-domain-system approach. in: Proc. SEKE'02. ACM, 2002, pp. 233-236
[125] WEINREICH, R., BUCHGEHER, G.: Integrating Requirements and Design Decisions in Architecture Representation. in: Babar, M.A., Gorton, I. (eds.): Proc. ECSA'10. LNCS, Vol. 6285. Springer, 2010, pp. 86-101
[126] XEOX, http://www.xeox.com, accessed 22.08.2012
[127] YAGER, R.R.: On Prioritized Multiple-Criteria Aggregation. IEEE Transactions on Systems, Man, and Cybernetics 2012, in press,
[128] YU, H., HATZIVASSILOGLOU, V.: Towards answering opinion questions: separating facts from opinions and identifying the polarity of opinion sentences. in: Proc. EMNLP'03. Assoc. for Computational Linguistics, 2003, pp. 129-136