Let Stakeholders Define Quality: A Model-Based Approach

Vladimir A. Shekhovtsov, Heinrich C. Mayr

Application Engineering Group, Institute of Applied Informatics,
Alpen-Adria-Universität Klagenfurt
Universitätstrasse 65-67, 9020 Klagenfurt, Austria
shekvl@yahoo.com, heinrich@ifit.uni-klu.ac.at

Summary. The paper describes the means of flexible model-based process support for continuous quality-related interaction between business stakeholders and software developers. The support is based on model-driven and situational method engineering paradigms. The support processes are defined as the results of enactment of the corresponding process models; before the enactment, these models are tailored to the task at hand by the specific adaptation process. The obtained solution allows the developers to be continuously aware of the stakeholders' opinions on quality of the prospective system.

1. Introduction

Our research is motivated by the problems of defining the software process activities of collecting the opinions of business stakeholders, in particular, their opinions on the quality of the prospective system (for our research, we selected performance and reliability as the relevant quality characteristics). The need to perform these activities continuously is supported by both theoretical work (the concept of *reaching out* of *bunkering* in [1] i.e. the need in breaking potentially damaging "submerging" into the development activities by obtaining the opinions of stakeholders on the work done so far) and the development practice (e.g. the regular check-ins in Scrum [20]).

There are problems with these activities that make their organization difficult:

1. It is difficult for stakeholders to express the opinions on the quality of the prospective system if they cannot experience it in the appropriate context. Without this, they can only formulate the imprecise opinions e.g. "the system must be reliable".
2. The process of assessing the quality of the prospective system depends on the anticipated or implemented interaction of this system with its environment; this assessment is a complex task.

To address these problems, the QuASE project (Quality-Aware Software Engineering) was established in cooperation with two local IT companies. It aims at process and tool support for quality-related stakeholder interactions in a software process

with an ultimate goal of facilitating intelligent analysis of quality-related issues in software development [24-26].

This paper describes the means of flexible process support for the activities established as a part of the project. We define the model-driven approach to the proposed process solution, where every process is implemented as an enactment of the particular process model; and describe special process tailoring these process models to the problem at hand following situational method engineering paradigm.

The paper is structured as follows. Section 2 describes the necessary background information. Section 3 describes the proposed solution structure, followed by the description of the related work and the conclusions.

2. Related work

Several approaches exist aiming at using quality to drive particular software engineering tasks or on the different development stages. Among these approaches are Quality-Driven Re-Engineering [27], QuadREAD project [18] targeting transitions between requirements engineering and architectural design, approaches targeting quality-driven development restricted to architectural design phase [12].

Closer to the goal of our approach are approaches aimed at making the entire software process driven by quality. Examples are [6, 9], model-driven techniques such as Quality-Driven Model Transformation [15], more recent model-driven techniques [4]. These approaches, however, do not envision organizing direct interaction with stakeholders.

3. Background information

3.1. Conceptualization of quality-related knowledge

Prior to defining the process support for quality-related stakeholder interaction activities, the knowledge of these activities needs to be acquired and conceptualized. We introduced the original approach for conceptualizing the software quality in [21] and extended it in [25]. This work expressed the notions of software quality and its usage in the terms of the formal ontology [14]. Another body of preliminary research was related to conceptualizing the knowledge about quality attributes and quality requirements. To solve this problem, we introduced Quality-Aware Predesign Model for Services [22, 23], for our purposes, we generalize it to cover activities not specific to software services.

Based on these conceptualizations, we are currently working at establishing a common ***StakeQPA*** ontology (the ontology of Stakeholder Quality Perception and Assessment). Describing this ontology in detail is out of scope of this paper (the categories of knowledge to be incorporated are enumerated in [26]); here we only state that it extends the conceptual notion of quality by defining, among others, the concepts of *Expected* and *Proposed quality* (the complete list of concepts is introduced in [25]).

The *Expected quality* is defined as an image of a quality of the system under development conceived of by the individual stakeholder as a quality he/she wants the final system's version to provide. This kind of quality needs to be taken into account when the software process deals with stakeholder's opinions, but it exists only in a stakeholder's mind. The *Expected quality* is an individual quality as opposed to the collective quality reflected in quality requirements.

The *Proposed quality* denotes a set of values for quantified quality attributes of the system under development reflecting the current state of its development and represented in a form suitable for the perception by stakeholders. It can be obtained from e.g. a simulation model of the qualities of the system under development reflecting its current state of development or the system prototype.

3.2. Continuous quality awareness

We focus on formalizing the process of interaction between business stakeholders and developers utilizing the obtained ontological knowledge. We plan to base this process on the idea of achieving *Continuous quality awareness* [24]. It means that the developers should be able to check both the desired quality of the software under development (SUD) from the point of view of its stakeholders and the current position of the SUD with respect to that quality throughout the software process. Achieving such awareness correspond to the notion of *reaching out* from [1], when the members of the development team are supposed to continuously break the "bunkering" barrier to make themselves aware on the stakeholders' opinion on the work done so far.

We need to formalize the set of awareness support processes to be performed continuously on different software process stages. They should provide the means of getting the correct stakeholder opinions on quality. For elaborating such processes, we use interactive simulation of software performance and reliability defined as follows:

- Quality simulations are parameterized by the factors that influence qualities; they are executed interactively in the context of the system usage;
- Business stakeholders experience *simulated* qualities in *simulated* usage contexts corresponding to their roles, and assess this experience;
- The obtained assessments are used as a basis for software engineering activities to achieve the desired level of quality awareness.

4. Proposed solution structure

4.1. Model-based process engineering activities

We propose to define the set of processes directing awareness support activities to be integrated into the common *Awareness Support Process*. For this task, we rely on model-driven and ontology-based software engineering paradigms by composing *process models* for the necessary activities based on StakeQPA knowledge; they, in turn, are to be integrated into the common process model*:* *Awareness Support Model*. Launching *Awareness Support Process* is the result of the enactment of this model.

Establishing the process support entails the following activities (Fig.1):

1. *Engineering core models.* To establish process models, we make use of the Situational Method Engineering paradigm [8] aimed at allowing for engineering such models [2, 28] tailored to the problem at hand out of the reusable process fragments [19]. We plan to formalize the set of models to serve as such fragments (*Core Models*) based on the StakeQPA knowledge.

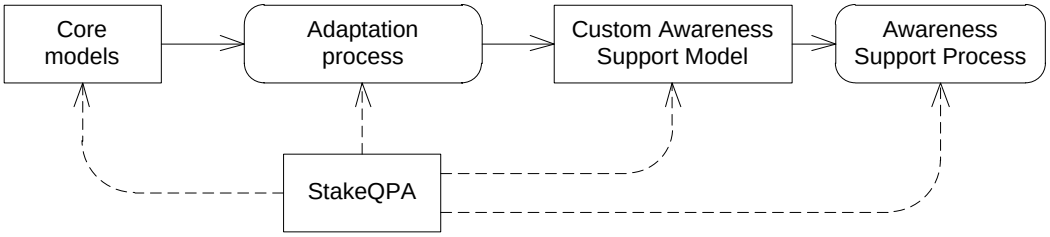

Fig.1. Model-based process engineering activities

2. *Establishing and formalizing adaptation process.* We formalize the *Adaptation Process* adapting the *Awareness Support Process* to the problem at hand. It contains the activities for (1) assembling the problem-specific set of process models to be incorporated into the *Awareness Support Model* while making use of the *Core Models*; (2) tailoring these models to the specific problem.
3. *Establishing and formalizing Awareness Support Process.* Based on StakeQPA and *Core Models*, we formalize the process models comprising the *Awareness Support Model*. The corresponding processes aim at addressing the goals of the project; both stakeholders and developers are supposed to interact with them.

In this paper, we concentrate on the activities of the adaptation process. More detailed treatment of the core models and the Awareness Support Process is presented in [24]; here we provide only necessary background information.

4.2. Engineering core models

To achieve the goals of QuASE adaptation to the problem at hand, we formalize the set of *Core Models* to be later combined by the *Adaptation Process* using the knowledge about the problem to form the custom *Awareness Support Model*. In this section, we describe two categories of such models aimed at organizing forming the *Proposed Quality* and revealing the *Excepted Quality* for particular qualities. Such models are used in adaptation mode to form the process models related to revealing all the qualities for the particular functional units of the system under development.

Quality Revealing Models formalize processes of organizing interactions with stakeholders to reveal the particular *Expected Quality* characteristics making use of StakeQPA ontological knowledge. The input for such process is the value of the *Proposed Quality*; its output is a revealed *Expected Quality* value obtained from a business stakeholder.

The implementation of the modeled process depends on the quality characteristic, stakeholder type, and project type. The interaction process can take a form of: (1) human-computer interaction; (2) human-to-human interaction (e.g. analyst delivers

the *Proposed Quality* value to stakeholder and asks for assessment); (3) empirical method-based interaction (using questionnaires etc.).

Enacting *Quality Revealing Model* creates the instance of the interaction process which (1) receives the numerical quality value as an input (not depending on its source), (2) makes the stakeholder experience this quality according to this value, (3) involves the stakeholder in an interaction related to an assessment of his/her experience and (4) returns the assessment value obtained from the stakeholder.

Quality Forming Models formalize processes of forming the *Proposed Quality* values. The particular implementation of such process depends on the particular quality characteristic and project type. Its goal is to present the *Proposed Quality* to the *Quality Revealing Process* (created as a result of the enactment of *Quality Revealing Model*). Its input is a set of values for factors influencing *Proposed Quality*. We plan to establish two categories of *Quality Forming Models*:

- Backed by real software solutions; for such implementation, we plan to formalize the adapter models defining processes of connecting to the running code with a goal of obtaining the *Proposed Quality* values;
- Based on simulation models; such implementations describe the processes for parameterized simulation of *Proposed Quality* values. These processes are responsible for composition and execution of simulation models.

4.3. Model-based adaptation process

We start from describing *Adaptation Process* (Fig.2) input and output artifacts. Among its necessary inputs are: the StakeQPA ontology, the set of core models, and the knowledge possessed by the software engineers involved into the adaptation. The output from this process will be *Awareness Support Model* representing *Awareness Support Process* tailored for the problem at hand. Below we enumerate its activities.

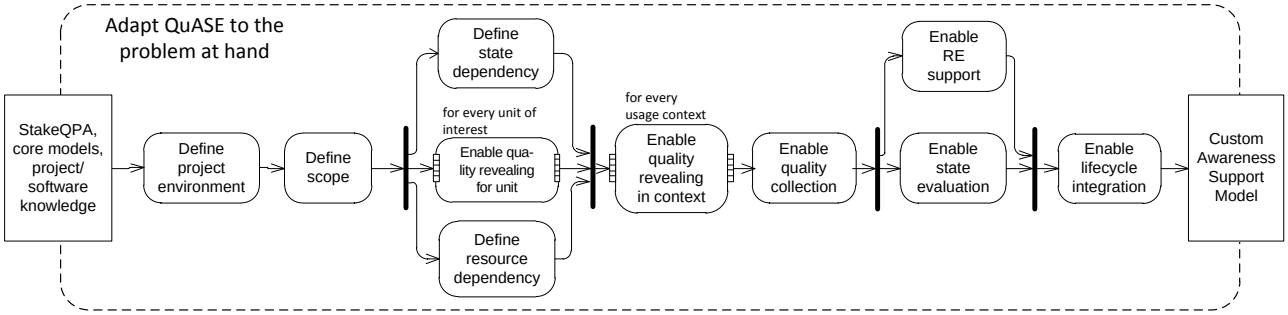

Fig.2. Adaptation Process

Define project environment tailors the *Awareness Support Process* to the task at hand by performing the following lower-level activities (1) forming the set of available categories of stakeholders, (2) categorizing and defining the properties of the organization under scrutiny, (3) defining the properties of interest for the project at hand. The *Project Environment Model* is an output of this activity.

Define scope sets the *Scope* of the SUD considered as the set of *Quality-Bearing Units* of interest defined for this system and the set of qualities of interest connected

to these units. The type of such units (e.g. services or service operations for the service-oriented system) and, therefore, the approach to decomposing the system into such units, depends on the system's type defined in *Project Environment Model* so this activity directly depends on the results of the environment definition activity. As a result, the *Scope Model* has to be produced.

Define state dependency models the set of dependencies between the description of the state of the system (e.g. software architecture or the particular architectural decision) and the factors influencing *Proposed Quality* (examples are e.g. [3, 7, 13]) for the problem at hand; producing the *State Dependency Model*.

Define resource dependency models the set of dependencies between the development resources and the factors influencing *Proposed Quality* for the problem at hand (an example of cost-involving dependency can be [16]) producing the *Resource Dependency Model*.

Enable quality revealing for a SUD unit aims at the composition of *Quality Revealing Model* and *Quality Forming Model* grouped by the *Quality-Bearing Unit* of interest included into the *Scope*. For example, for the service-oriented system we can group qualities by service or separate service operation. The result of this activity is a *Unit-Level Model* for a particular unit (e.g. service operation). Its enactment should be able to make the business stakeholder experience the *Proposed Quality* and reveal the *Expected Quality* for the particular quality-bearing unit. *Adaptation Process* performs this activity for all units of interest defined by *Scope Model* to obtain the set of *Unit-Level Models* corresponding to the *Scope*.

Enable quality revealing in context provides particular system under development's usage context [10] to ensure realistic stakeholder experience. We define such context as the way of organizing a meaningful interactive session between the stakeholder and the enactments of the *Unit-Level Models* for the *Scope*. In [11] we proposed to represent such context by a *Context-Level Model* defining an interactive process allowing stakeholders to participate by describing the sequence of actions for the particular stakeholder interaction session. It embeds *Unit-Level Models* corresponding to unit-level interactions composed for every *Quality-Bearing-Unit* of interest. *Adaptation Process* performs this activity for all the usage contexts to obtain the set of necessary *Context-Level Models*.

Enable quality collection models a process of controlling different *Expected Quality*-revealing sessions in different contexts and the integration of the results of these sessions obtaining the *Quality Collection Model*. With its enactment, it should be possible to execute such sessions for the selected contexts in the selected order, collecting both formed *Proposed Quality* and revealed *Expected Quality* from these sessions.

Enable requirements engineering support establishes the *Requirement Engineering Support Model* aimed at using the obtained *Expected Quality* to get quality awareness at the requirement engineering stage. Its enactment should allow:

- Eliciting new quality requirements (defined as e.g. "the quality value for the *Quality-Bearing Unit q* must exceed the *threshold*") by defining the corresponding

threshold values e.g. by aggregating them out of the captured output of the *Expected Quality* collecting activity comprising both *Expected Quality* and *Proposed Quality* values. For this aggregation, it should take into account such information as the relative importance of stakeholders or contexts etc.

- Validating already-existing quality requirements by comparing them to the revealed *Expected Quality* values.

Enable state evaluation defines the *State Evaluation Model* to be enacted to evaluate the particular state of the system under development (snapshot evaluation mode) or the assessments of the influence of the particular state-modifying decisions (incremental evaluation mode) by forming *Proposed Quality* values corresponding to the particular state of the system (or the change in this state) and collecting *Expected Quality* values representing the stakeholders' opinions on the state in question.

Enable lifecycle integration aims at establishing the *Lifecycle Integration Model* aimed at integrating obtained awareness support data into the software lifecycle to achieve continuous quality awareness. The integration itself can be implemented e.g. by defining the notion of *Quality-Related Issue* (in the same sense as used in the issue-tracking systems such as JIRA or Mantis) raised on different software process stages. Handling such issue could utilizes the information from the *Lifecycle Integration Model* enactment process and compares this information with e.g. the *Proposed Quality* values obtained from the *Resource Dependency Model* enactment process.

The information about past *Quality-Related Issues* (supplemented by rich semantics obtained from StakeQPA) need to be stored into the *Semantic Issues Repository*. This repository is planned to support intelligent analysis of such issues such as predicting the behavior of the stakeholders based on the issues encountered in the past [26].

4.3.1. Implementation support

To facilitate integration of the proposed adaptation solutions into the software process, we plan to make use of the capabilities of the ***Eclipse Process Framework*** (EPF): we plan to describe the set of activities according to the SPEM metamodel [29] and establish this representation in EPF. In addition, following [17], we aim at elaborating a Computer-Aided Method Engineering (CAME) solution aimed at designing custom interaction-supporting process models out of abstract method components, configuring these models with concrete technological assets (such as Eclipse-based model editors) and generating custom interaction-supporting CASE tools.

4.4. Awareness support process

Detailed description of the *Awareness Support Process* can be found in [24], here we emphasize its main features (Fig.3). Its inputs are the *Adaptation Process*-customized *Awareness Support Model* and the knowledge about the quality-aware decision to make. Its output is the information necessary for making a quality-aware decision.

The *Awareness Support Process* is started as a result of encountering the *Lifecycle Integration Model*-defined point of interest in a software process (launching a *Quali-*

ty-Related Issue). Then, the system performs the necessary actions to start the specific awareness support activity for this issue. In parallel, software engineer is asked to specify the information specific for the context of the solution at hand (such as the values of quality-influencing factors), this activity is called *QuASE parameterization*.

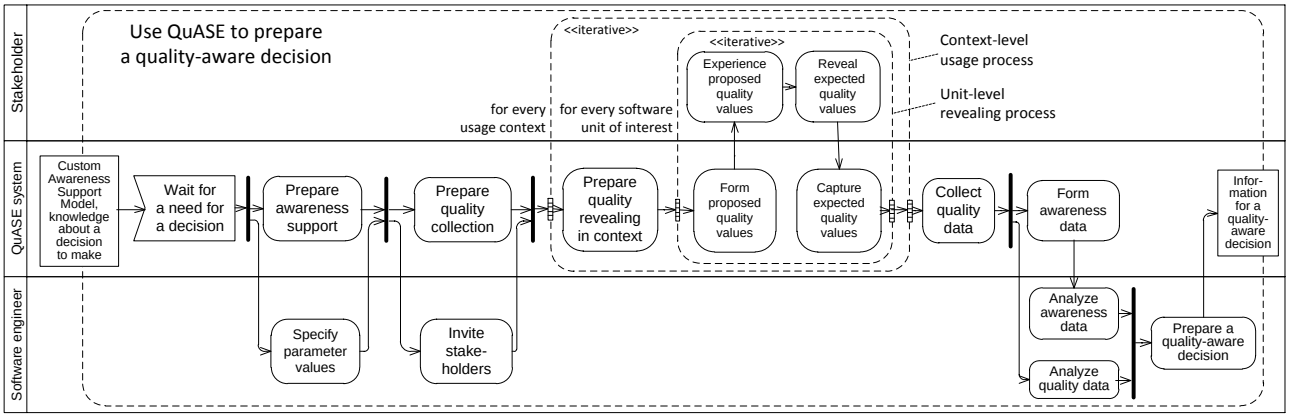

Fig.3. Awareness Support Process (from [24], with modifications)

To reveal *Expected Quality* in contexts, *Quality Collecting Model* enactment process iteratively enacts the set of *Context-Level Models* for all the selected contexts launching the interactive *Quality Revealing Processes* ready for stakeholders to participate. Every such process is presented to the stakeholder belonging to the particular role. During the run, when the logic of the usage process workflow requires invoking an activity representing the *Quality-Bearing Unit* of interest, the forming of *Proposed Quality* (e.g. by simulation) and the revealing of *Expected Quality* are handled by the corresponding *Unit-Level Model* enactment process. The outputs for the run include the set of revealed *Expected Quality* values (e.g. the assessment results).

After some of all revealing interactions are completed, the system collects the values for both *Proposed Quality* and *Expected Quality* from the results of these interactions. Then, the enactment process for *Requirements Engineering Support Model* or *State Evaluation Model* forms the necessary awareness data out of these collected values. This data can be then directly or indirectly used by the software engineers in the process of making the quality-aware decisions.

5. Conclusions

The proposed flexibility-enabled solution is justified by the fact that both application developers and business stakeholders are expected to get benefits from it.

Application developers will benefit from (1) a possibility for getting access to the core models as reusable method components making possible building customized quality-aware software processes tailored to the problem at hand; (2) a support for continuously taking into account the opinions on quality expressed by stakeholders. These benefits can serve as foundations for a competitive advantage of consistently delivering the software with stakeholder-expected quality.

Business stakeholders will benefit from a solution aimed at making them immune to their real preferences related to the quality of the prospective system getting mishan-

dled (e.g. misunderstood or lost); this is guaranteed by the processes of awareness support and continuous lifecycle integration of such support. These factors establish the ground for increasing the degree of stakeholder satisfaction.

From the economical point of view, establishing the foundations for achieving the flexibility of the QuASE solution allows for better cost management advantages compared to other solutions. For example, even the most complete process-based solutions such as [5] are built in top-down fashion, as a result, complete process model needs to be built in any case; other techniques are even less flexible.

Literature

[1] Adolph, S., Kruchten, P.: Reconciling Perspectives: How People Manage the Process of Software Development. In AGILE'11, 48-56, IEEE, 2011

[2] Becker, J., Janiesch, C., Pfeiffer, D., Seidel, S.: Evolutionary method engineering: towards a method for the analysis and conception of management information systems. In Proc. 12th Americas Conference on Information Systems (AMCIS 2006), 3686-3697, 2006

[3] 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, 86-102, Springer, 2008

[4] de Miguel, M.A., Massonet, P., Silva, J.P., Briones, J.: Model Based Development of Quality-Aware Software Services. In ISORC'08, 563-569, IEEE, 2008

[5] Fritzsche, M., Gilani, W., Fritzsche, C., Spence, I., Kilpatrick, P., Brown, J.: Towards utilizing model-driven engineering of composite applications for business performance analysis. In ECMDA-FA'08, 369-380, 2008

[6] Grambow, G., Oberhauser, R., Reichert, M.: Contextual Injection of Quality Measures into Software Engineering Processes. In Int'l Journal on Advances in Software, 4, 76-99, 2011

[7] Grassi, V., Mirandola, R., Sabetta, A.: Filling the gap between design and performance/reliability models of component-based systems: A model-driven approach. In The Journal of Systems and Software, 80, 528–558, 2007

[8] Henderson-Sellers, B., Ralyte, J.: Situational Method Engineering: State-of-the-Art Review. In Journal of Universal Computer Science, 16, 424-478, 2010

[9] 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, 137-144, ACM, 2010

[10] ISO 9241-210:2010. Ergonomics of human-system interaction - Part 210: Human-centred design for interactive systems. International Organization for Standardization, 2010

[11] 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, 135-140, Springer, 2008

[12] Kim, S., Kim, D.-K., Park, S.: Tool support for quality-driven development of software architectures. In ASE'10, 127-130, ACM, 2010

[13] Marzolla, M., Balsamo, S.: UML-PSI: the UML Performance SImulator. In Proc. QEST'04, 340-341, IEEE, 2004
[14] Masolo, C., Borgo, S.: Qualities in formal ontology. In Proc. Workshop on Foundational Aspects of Ontologies (FOnt 2005), 2-16, 2005
[15] Matinlassi, M.: Quality-Driven Software Architecture Model Transformation. In Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture, 199-200, IEEE Computer Society, 2005
[16] Mutschler, B., Reichert, M.: Exploring the Dynamic Costs of Process-aware Information Systems through Simulation. In EMMSAD'07, 163-172, 2007
[17] Paulk, M.C.: Key Practices of the Capability Maturity Model, Version 1.1, technical report CMU/SEI-93-TR-025. Software Engineering Institute, Carnegie Mellon University, 1993
[18] Project QuadREAD, http://quadread.ewi.utwente.nl, accessed Mar. 26, 2012
[19] Ralyte, J., Deneckere, R., Rolland, C.: Towards a generic model for situational method engineering. In CAiSE 2003, LNCS, Vol. 2681, 95-110, Springer, 2003
[20] Schwaber, K., Beedle, M.: Agile Software Development with Scrum. Prentice Hall PTR,, 2001
[21] Shekhovtsov, V.A.: On the evolution of quality conceptualization techniques. In The Evolution of Conceptual Modeling, LNCS, Vol. 6520, 117–136, Springer, 2011
[22] Shekhovtsov, V.A., Kaschek, R., Kop, C., Mayr, H.C.: Relational service quality modeling. In Non-Functional Properties in Service Oriented Architecture: Requirements, Models and Methods, 172-193, IGI Global, 2011
[23] Shekhovtsov, V.A., Kop, C., Mayr, H.C.: Capturing the Semantics of Quality Requirements into an Intermediate Predesign Model. In SIGSAND-EUROPE'2008 Symposium, LNI, Vol. P-129, 25-37, GI, 2008
[24] Shekhovtsov, V.A., Mayr, H.C., Kop, C.: Stakeholder Involvement into Quality Definition and Evaluation for Service-Oriented Systems. In Proc. USER'12 Workshop at ICSE'12, 49-52, IEEE, 2012
[25] Shekhovtsov, V.A., Mayr, H.C., Kop, C.: Towards Conceptualizing Quality-Related Stakeholder Interactions in Software Development. In Accepted for publication in UNISCON 2012, LNBIP, Springer, 2012
[26] Shekhovtsov, V.A., Mayr, H.C., Kop, C.: Acquiring Empirical Knowledge to Support Intelligent Analysis of Quality-Related Issues in Software Development. In QUATIC'12, IEEE, 2012, in press
[27] Tahvildari, L., Kontogiannis, K., Mylopoulos, J.: Quality-Driven Software Re-engineering. In J. of Systems and Software, 66, 225–239, 2003
[28] van de Weerd, I., Brinkkemper, S.: Meta-modeling for situational analysis and design methods. In Handbook of Research on Modern Systems Analysis and Design Technologies and Applications, 35-54, IGI Global, 2009
[29] Wada, H., Suzuki, J., Oba, K.: A Model-Driven Development Framework for Non-Functional Aspects in Service Oriented Architecture. In International Journal of Web Services Research, 5, 1-31, 2008