A Threefold User-Based Software Evaluation Framework

Radu Vlas
Georgia State University, rvlas@cis.gsu.edu

Cristina Vlas
Omnis Prime, cvlas@omnisprime.com

A Threefold User-Based Software Evaluation Framework

Radu Vlas
Georgia State University
rvlas@cis.gsu.edu

Cristina Vlas
Omnis Prime
cvlas@omnisprime.com

ABSTRACT

While pressured to release new software to market, software development organization often find themselves choosing between quality and timeliness. From a business perspective, success is determined to a significant extent by the perceptions of quality exhibited by consumers with respect to the software product being offered. In this context, we propose taxonomy of software users and design a user-centered software evaluation framework to support software quality assessment processes. An online survey is conducted for exploring software users' perceptions of quality and the results are used to define three categories of software users. The findings of this study shed new light on the way software development organizations should perceive the relationship with their consumers and, consequently, on the way they should manage development projects. The contributions to both research and practice stem from the proposing of a new and more detailed approach to defining and measuring software system quality and its associated assessment processes.

Keywords

IS evaluation, IS quality, theory of emotions, emotion process model, user perspective, over-subjectivity.

INTRODUCTION

Software engineering domain represents a relatively young area of research when compared to social or natural sciences. Interdisciplinary research linking between areas of software engineering and social sciences is still limited in both number and complexity. Thus, it comes as a logical consequence that a thorough understanding of some areas of software engineering has not yet been achieved. Software product quality represents one such area in which there are continuous research efforts to crystallize a convergent perspective. Over time, researchers and practitioners have attempted to define software quality through the lens of numerous views: product-based view, manufacturing-based view, user-based view, or various other views (Wong 2002). For the scope of this study, it is important to note that only few software quality models include users' perspectives on quality into the overall concept of software quality. A comprehensive model of users' perspectives on software quality has not yet been developed. This research posits that one's emotional formation, technology-related knowledge, and past experiences determine one's software quality assessment strategy and the associated outcomes. We investigate a set of factors generating various user perspectives and we propose a software quality assessment framework.

The importance of software quality has been highlighted by Broy in 2004: "software quality is a crucial issue in a society that vitally depends on software systems" (Broy, Deißenböck and Pizka 2004). In spite of this fact, software products being released to market show a large number of weaknesses (Rao and Monroe 1988; Arora, Caulkins and Telang 2005; Newsham, Palmer, Stamos and Burns 2007). Research associates this fact with an increasing complexity of software systems (Offutt 2002). Another explanation indicates that the tendency of releasing a less functional software product earlier (first to market) (Bayus 1997) is justified by the limited amount of resources required later for fixing it (Arora et al. 2005). It has been shown that lower software quality has a negative impact on the market, the users, and the releasing company's market valuation (Choudhary 2007). In spite of all these negatives, Banker and Slaughter confirm the economical viability of a "first to market" approach (Banker and Slaughter 1997). The question this raises is, *while software development organizations maintain a "first-to-market" strategy, what areas of development should receive additional support in order to maximize software users' perceptions of quality?* Therefore, this study attempts to define and describe the existing categories of individual software users along with their perceptions of quality. Adoption decisions within organizations are often discussed and reached at a group level rather than at an individual level. In spite of this, the study remains focused on individual users since the existing literature on group behaviors points to the significant impact of individual emotional features on group behaviors (Barsade 2002).

THEORETICAL BACKGROUND

As described by a wide range of researchers, quality is a measure of conformance. Software quality has been defined as either conformance to users' needs and expectations, or conformance to specifications. Traditionally software quality has been defined through characteristics of the software development processes (Nord and Tomayko 2006), through characteristics of the software product itself (Plosch, Gruber, Hentschel and Korner 2007), or through attributes emphasized by the users of the software product (Feigenbaum 1983; Ishikawa 1985). These last three perspectives form the basis for a large number of generally accepted software quality philosophies and models (Berander, Damm and Eriksson 2005). In spite of this, there is still a need for an articulated view delineating the role the users play in assessing software quality. The users' perceptions of quality and the individual assessment criteria they use were never formalized into a quality assessment methodology.

Although there is a large number of quality standards and models available, practitioners tend to most often assess quality based on the two standards the International Organization for Standardization published. ISO/IEC 9126 ((ISO) 2001) has a four part structure that addresses the following areas: process quality (quality of the development process), internal quality (quality of the source code and architectural quality), external quality (quality delivered by the software product, quality of execution), and quality in use (to which extent user needs are met in the user's working environment). ISO 25000 includes requirements specification into the quality evaluation process (Zubrow 2004).

New perspectives bring to attention new factors that complement the ISO standards. According to Armour, software is more a medium for storing knowledge than a product, while a software development process is more a knowledge generation process than a product manufacturing process (Armour 2000b, a). While acknowledging Armour's view, one has to accept that the general definition of software product quality provided by the ISO standards should be complemented with a human-centric view in which personal traits play a significant role.

The theory of emotions (Frijda 1986) provides a means of including personal traits into the software quality evaluation process. Starting from it, we develop a software quality assessment framework which is described in the next sections. The emotion process model (Frijda 1996, 2007) describes the instantaneous emergence and transformation of emotion components at the individual level. Behaviors are exhibited as a response to a set of individual emotions. Individual emotions are shaped by one's personal reaction to the results of appraising a current condition or event.

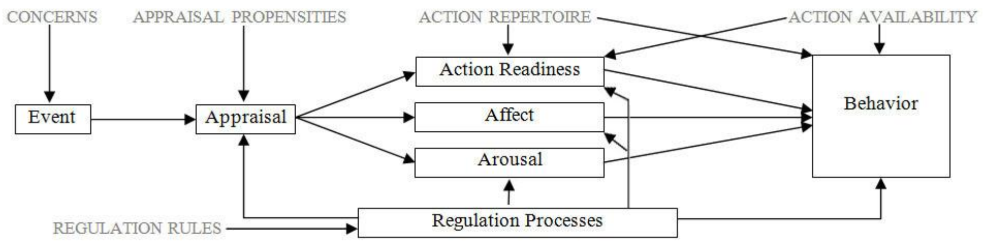

Figure 1: The Emotion Process Model and Its Condition (Frijda 1996)

In the context of software quality assessment, *current condition* is defined by an individuals' exposure to a software artifact. *Appraising* refers to "the information processes that link perception of an event to emotional meaning" (Frijda and Mesquita 1998). During the appraisal process, one will consider the potential use of the software artifact as a means of addressing personal concerns. The intensity of emotions elicited during the appraisal process is correlated with the strength and the importance of the individual *concerns* addressed by the condition being appraised. *Affect*, *arousal*, and *action readiness* refer to an individual's emotional processes generating the preconditions for a behavior. *Regulation processes* refer to the attenuation or enhancement of emotion because of anticipated effects. The two types of regulation processes, internal (from inner subject), and external (from the environment) have a significant influence on one's quality assessment (Frijda 1986; Frijda and Mesquita 1998). Internal regulation processes can be personal preferences, individual norms and standards, or pre-determined reactions formed as a consequence of previous experiences or acquired knowledge. One's past experiences might have a unidirectional influence on one's emotion processes. If the person has knowledge in a relevant area then the strength and direction of emotions will be influenced by the match between the artifact's expected and perceived behavior. External regulation processes include societal, organizational, or group norms and standards. An individual might be externally constrained to perceive an artifact as not being beneficial or efficient. Some people believe that Microsoft products are unreliable. A person sharing this belief can assign estimates of quality before confirming such assessment. *Behavior* includes intentional behavior that is motivated by action tendency. People with different experiences, knowledge, and understanding of software systems develop different sets of emotions and, therefore, assess the quality of software differently.

The proposed software evaluation framework is intended to be used for assessing the quality of a software system and, consequently, for determining the expected success of the software project. Thus, the proposed framework is presented as an artifact that delineates specific categories of individual perceptions of quality and provides a means to systematically analyze these perceptions in a consistent framework. With this perspective in mind, we consider the technology acceptance model (TAM) to provide a larger scale, more general view of the same perspective (Davis 1989). TAM emphasizes the importance of individual perceptions in the decision making process. TAM posits that a person's intention to use a software artifact is determined by a set of perceptions that person has (Venkatesh 2003). An individual's perceptions of usefulness and ease of use are significantly influenced by that individual's past experiences and knowledge in areas relevant to the software system. For instance, a data mining expert might decide not to use a software tool based on two factors: first, the data mining expert possesses the knowledge required to manually perform the tasks at hand; second, the data mining expert recalls a negative experience from the past while using a similar software tool. Therefore, our data mining expert might decide (e.g. assess tool, or use tool) largely based on past experiences and available knowledge.

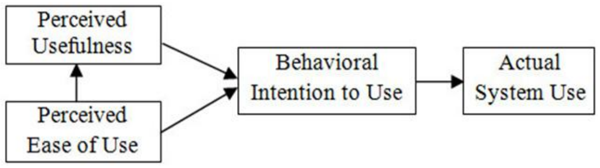

Figure 2. The Technology Acceptance Model (Davis 1989)

A USER-CENTRIC SOFTWARE EVALUATION FRAMEWORK

A taxonomy of software user perspectives

Humans' reactions to events and situations they face are based on personal traits. A person's past experiences influence that person's emotions associated with the appraisal of a current context and that person's decisions regarding aspects of technology. Therefore, past experiences can become a determinant factor in choosing a specific course of action in a given situation. Also, a factor with a significant influence on one's decision process is the knowledge relevant to the decision process. For example, if an IT manager has to choose what web browser to install on a small organization's network, then s/he will use the knowledge acquired over time for deciding which browser would provide most benefits while exposing organization's network to less risk. This is not the case of a user with limited IT-related knowledge. This user will largely use interface and functionality related attributes as main criteria for making a decision. Therefore, people with advanced technical knowledge and experiences tend to use different, and usually larger, sets of criteria when assessing the quality of a product or service. While the quality assessment approaches adopted by people with and without advanced technical knowledge and experiences are different, they also overlap in few respects. All software users, regardless of background, agree that ease of use is a desired property of software. In spite of this overlapping, the two opposite perspectives use distinct approaches. We use this distinction for defining the scope of our taxonomy of users of software. We distinguish our study from previous research by focusing on assessment approaches instead of on perceptions of quality alone (Jung, Kim and Chung 2004).

We group common quality factors into seven sets of criteria to better understand the taxonomy of software users, ranging from technical to basic, and to define their respective approaches to quality. These sets of criteria are: (1) general measures of quality, (2) complementary services and components, (3) individual preferences, (4) subjective measures, (5) front-end specific measures, (6) back-end specific measures, and (7) features. These sets are detailed in the next paragraphs. In subsequent sections we design and propose a user-centric software evaluation framework including these 7 sets of quality factors. We claim that first two sets include general quality factors commonly shared by all software users. Next two sets include factors which are more specific to the non-technical side of our taxonomy. Last three sets include factors only a user with technical background would completely understand.

The elements common across our taxonomy are captured in the general measures of quality and in the complementary services and components. General measures of quality include measures that are generally accepted by all software stakeholders as indicators of quality. Such measures are performance (e.g. speed), efficiency (e.g. level of computer resources utilization), accuracy (e.g. degree of answer correctness), functionality (e.g. customization, solution finding algorithm, user-system interaction dynamics), reliability (e.g. error ratio in a given context and over a determined period of time), robustness (e.g. ability to recover from unexpected situations or improper functionality parameters), complexity (e.g. user perceived level of complicatedness), and other similar generally accepted measures of quality. Complementary services and components include documentation availability and completeness, maintenance support, and customer service support. While various approaches might differ in the way they analyze documentation, we consider the difference to not be significant.

Subjective measures of quality, which we consider to be specific to the non-technical software user, might include attachment (e.g. affection towards specific characteristics of a software artifact), or familiarity (e.g. knowledgeability of a software system's functionality and use due to previous experiences with similar artifacts). Last of the four sets of criteria a basic software user considers for assessing quality is represented by individual preferences. We split them up into functionality-related and presentation-related preferences. Functionality-related individual preferences might refer to one's liking of drop-down menus as opposed to pop-up options, selection lists, or wizard-type selection when facing multiple options. Presentation preferences are related to various aesthetic elements (e.g. graphical elements used to highlight frames in a multi-frame interface), choice of colors or their combination in a color scheme, or structure and organization of the graphical user interface (e.g. grouping of similar elements of a GUI in the same area of the interface or consistent look and feel).

Technical software users' perspective includes the two sets of quality measures shared with the basic software users' perspective and three additional sets of measures. First of the additional sets refers to the features the software system provides. One can assess the customization level of the software system (e.g. the extent to which the software system is able to offer customization options), completeness (e.g. the extent to which the software system properly addresses the complete set of user needs and expectations), flexibility (e.g. the ability to adapt to various types of functionality requirements and sets of inputs), modularity (e.g. the organization of the system into a structure of inter-dependent and communicating modules). Last two additional sets relate to front-end and back-end elements of a software system. Front-end measures of quality are related to the user interface. A person with advanced knowledge related to standards of GUI design can use Miller's law as a way of confirming or disconfirming a software system's quality. This is not the case of a basic user lacking such knowledge. Back-end measures of quality address characteristics relating to source code (e.g. indentation, comments, clarity, structure, choice of programming language, size), architecture (e.g. organization of functional units of code, choice of database support), database design and data communication (e.g. structure and organization of data in a database, data communication architecture and protocols), or software development life-cycle (e.g. development processes and methodologies employed).

A user-based software evaluation framework (USEF)

ISO 25000 presents the "product quality measurement reference model" in order to explain the steps along a product's lifecycle at which specific quality measures can be performed (software development process, internal quality, external quality, quality in use). Requirements specification is considered to be part of the process quality step. Given that software products have the specifics of knowledge mediums and since the software development activity matches the unique characteristics of a knowledge creation process, as suggested by Armour, we propose the explicit representation of requirements analysis and specification in the measurement reference model (Armour 2000b, a).

The ISO 25000 standard labels quality measures that are perceived by users as "quality in use." One's perception of quality is not strictly limited to characteristic directly derived from the software system itself. Often, the availability of maintenance options or support is considered an indication of quality, despite these not being integral part of the software system. Therefore, we add complementary services as a sixth component to the product quality model (Figure 3).

Those having an advanced technical background had the ability to look "deeper" into the characteristics of a software system, enabling them to exhibit a distinct approach to quality assessment. We posit that the level of experience and the amount of knowledge determines the software evaluation strategy used. Consequently, there is a wide spectrum of software users, ranging from technical users (extensive software knowledge and experience) to basic users (very limited knowledge and experience). In this study, we classified software users as technical, basic, or knowledgeable (users with certain knowledge and experience but without a deep understanding of technology). While technical users commonly include all six evaluation areas into their assessments, other users tend to place less importance on few evaluation areas. The framework we propose can be used for guiding the computing of "quality scores" corresponding to the three perspectives in our taxonomy (technical, knowledgeable, and basic). Quality scores are based on a set of weights highlighted in research data. The weights correspond to the importance of each evaluation area according to each one of the three perspectives. Technical users place a significant amount of importance on all six areas of the development lifecycle. In contrast, basic users assess quality based on only 3 main areas of the product lifecycle.

METHODOLOGY, RESULTS, AND EVALUATION

We use an online survey to collect data from students at a large-size university in the South-East United States and from software professionals at a Mideast branch of a global IT company. The survey results validate the proposed taxonomy of software users and support future efforts aimed at validating the USEF. Framework validation is only partially achieved in this paper through its development based on concepts established in the academic literature, and through logical reasoning. A total of 64 complete responses were received. Subjects were asked to rate their knowledge and experience level for 15 topics mapped to the 6 areas of the USEF, as well as to rate their perceptions of importance of a number of 13 criteria mapped to the 6 areas of the USEF. Subjects' answers to background questions are coded from 1 to 5, with average scores per USEF area indicating the type of user for that specific area (less than 2.5 - basic user, between 2.5 and 3.5 - knowledgeable user, over 3.5 - technical user). Since people might have advanced knowledge and experience in one area of software and less advanced knowledge in another area, we re-classify each subject for each evaluation area of the USEF.

The survey includes open-ended questions intended to capture detailed individual perceptions and knowledge related to the 6 evaluation areas of the USEF. The open-ended questions and their corresponding qualitative answers help us triangulate and validate results. Through the analysis of these qualitative answers we identify situations of over-subjectivity. For instance, for area 1, one technical user and three knowledgeable users rated importance as high or very high but provided details indicating they lack the knowledge and experience to completely understand area 1. These findings indicate the presence of over-subjectivity in the pool of subjects and dictate the need for triangulation of results (Rao and Monroe 1988). Over-subjectivity is the extent to which users believe they have knowledge they actually don't. Over-subjectivity might be a result of familiarity with the expression or concept used to describe the evaluation area (Park and Lessig 1981; Alba and Hutchinson 1987). For instance, the term "requirement" seems familiar to many knowledgeable users who don't actually have significant experience or knowledge relevant to requirements processes (gathering, verification, specification, management).

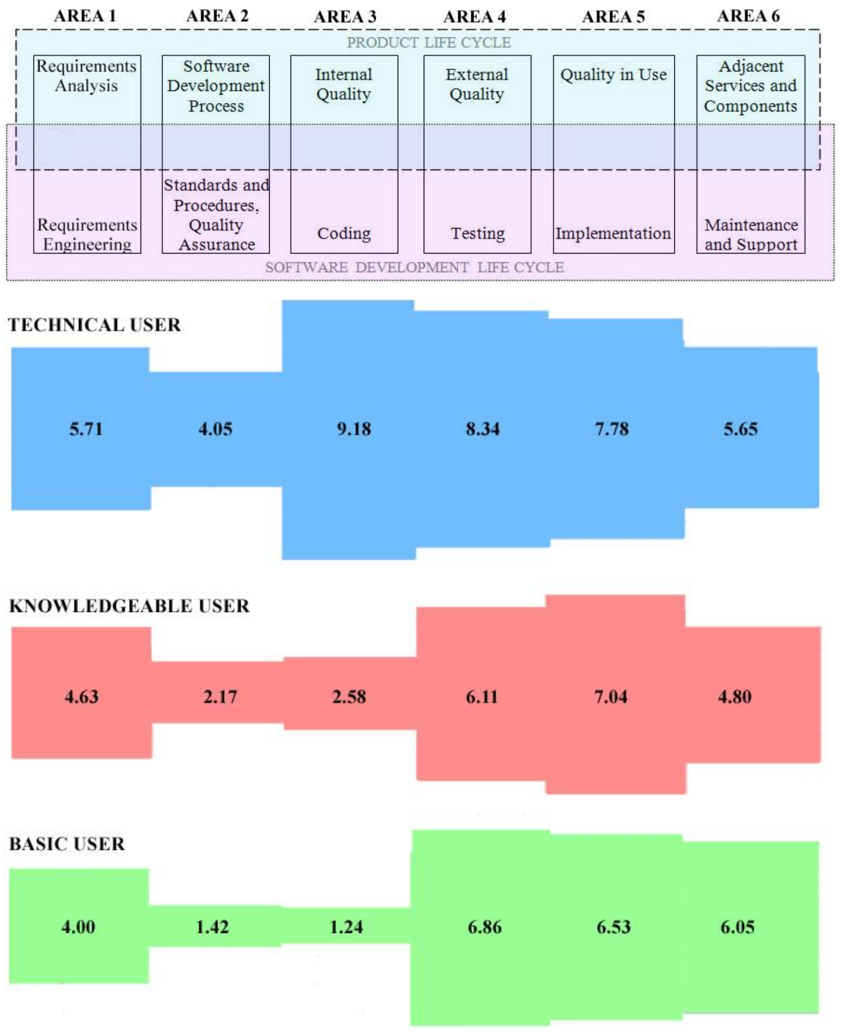

Figure 3. A User-Based Software Evaluation Framework and the Average Evaluation Area Importance (adapted from ((ISO) 2001; Zubrow 2004))

The perceived importance of evaluation areas is rated on a four equal-interval scale, and coded to numerical values between 0 and 10 (0-3.33-6.67-10). For each evaluation area, we compute average importance scores per category of users (Figure 3). Technical users place more importance than all other categories of users on all evaluation areas, except area 6 where basic users consider it more important. This can easily be explained by these two categories specific needs for maintenance. The most significant difference between technical users and basic/knowledgeable users is exhibited by the importance of internal quality (coding) – area 3. Technical users rate the importance of area 3 to the assessment of the overall quality of the software system as 9.18 out of 10. For the same area, basic users' average rating is only 1.24. This is explained by the knowledge and experience possessed by these two categories of users in this technical area. Similarities between knowledgeable and basic users' perceptions of evaluation area importance suggest that many knowledgeable users exhibit over-subjectivity. For example, for one question related to source code modularity, one knowledgeable user out of eleven provides a correct technical answer, while other eight of them consider this attribute to be of no importance and thus don't provide explanations. For same question, there are two technical users and both rate the importance as very high and provide detailed and accurate technical explanations.

CONCLUSIONS AND FUTURE RESEARCH

In this paper we propose taxonomy of software users and a user-based software evaluation framework. Using an online survey, we perform a comparative analysis of categories of software users, and perform the first steps required for completely validating the proposed framework. We find that technical consumers of software use their experience and knowledge in order to assess the quality of a software system significantly more than knowledgeable or basic consumers do. Basic users and especially knowledgeable users over-rate the importance of a number of quality assessment criteria due to their over-subjectivity. Basic users place a great importance on the availability of appropriate support and maintenance. Future research includes the complete validation of the proposed framework, the design of a software quality metric consistent with the proposed framework, and a larger sample of users. For validation purposes, we consider adding experimental research methods to the pool of techniques employed as they better fit the specifics of this study.

REFERENCES

1. (ISO), I. O. f. S. (2001). ISO/IEC 9126-1: Software engineering-product quality-part 1: Quality model. Geneva, Switzerland, International Organization for Standardization.
2. Alba, J. W. and J. W. Hutchinson (1987). "Dimensions of Consumer Expertise." Journal of Consumer Research **13**.
3. Armour, P. G. (2000a). "The Case for a New Business Model - Is software a product or a medium?" Communications of the ACM **43**(8): 19-22.
4. Armour, P. G. (2000b). "The Five Orders of Ignorance." Communications of the ACM **43**(10): 17-20.
5. Arora, A., et al. (2005). "Sell First, Fix Later: Impact of Patching on Software Quality." Management Science.
6. Banker, R. and S. Slaughter (1997). "A Field Study of Scale Economies in Software Maintenance." Management Science **43**(12): 1709-1725.
7. Barsade, S. (2002). "The Ripple Effect: Emotional Contagion and Its Influence on Group Behavior." Administrative Science Quarterly(47): 644-675.
8. Bayus, B. (1997). "Speed-to-market and new product performance trade-offs." Journal of Product Innovation Management **14**: 485-497.
9. Berander, P., et al. (2005). Software quality attributes and trade-offs, Blekinge Institute of Technology.
10. Broy, M., et al. (2004). A Holistic Approach to Software Quality at Work. World Congress for Software Quality.
11. Choudhary, V. (2007). "Software as a service: Implications for investment in software development." Journal of Management Information Systems **24**(2): 141-165.
12. Davis, F. D. (1989). "Perceived usefulness, perceived ease of use, and user acceptance of information technology." MIS Quarterly **13**(3): 319-339.
13. Feigenbaum, A. V. (1983). Total quality control, McGraw-Hill.
14. Frijda, N. H. (1986). The Emotions. New York, Cambridge University Press.
15. Frijda, N. H. (1996). Passions: Emotion and Socially Consequential Behavior. Emotion: Interdisciplinary Perspectives. R. D. Kavanaugh, B. Zimmerberg and S. Fein. (Eds.), Mahwah: Lawrence Erlbaum**:** 1-27.
16. Frijda, N. H. (2007). The Laws of Emotion, Mahwah: Lawrence Erlbaum Associates.
17. Frijda, N. H. and B. Mesquita (1998). The Analysis of Emotions: Dimensions of Variation. What Develops in Emotional Development? M. Mascolo and S. Griffin. (Eds.) New York, Plenum Press**:** 273-295.
18. Ishikawa, K. (1985). What is total quality control? : the Japanese way, Prentice-Hall.
19. Jung, H., et al. (2004). "Measuring software product quality: A survey of iso/iec 9126." IEEE Software **21**(5): 88-92.
20. Newsham, T., et al. (2007). Forensics software: Weaknesses in critical evidence collection. Proceedings of the 2007 Black Hat Conference.
21. Nord, R. L. and J. E. Tomayko (2006). "Software Architecture-Centric Methods and Agile Development." IEEE Software **23**(2): 47-54.
22. Offutt, J. (2002). "Quality Attributes of Web Software Applications." IEEE Software **19**(2): 25-32.
23. Park, K. W. and V. P. Lessig (1981). "Familiarity and Its Impact on Consumer Decision Biases and Heuristics." Journal of Consumer Research **8**: 223-230.
24. Plosch, R., et al. (2007). The EMISQ Method - Expert Based Evaluation of Internal Software Quality. Proceedings of 3rd IEEE Systems and Software Week, Baltimore, IEEE Computer Society Press.
25. Rao, A. R. and K. B. Monroe (1988). "The Moderating Effect of Prior Knowledge in Cue Utilization in Product Evaluations." Journal of Consumer Research: 253-264.
26. Venkatesh, V., Morris, M. G., Davis, G. B., & Davis, F. D (2003). "User acceptance of information technology: Toward a unified view." MIS Quarterly **27**(3): 425-478.
27. Wong, B. (2002). The Appropriateness of Gutman's Means-End Chain Model in Software Evaluation. Proceedings of the 2002 International Symposium on Empirical Software Engineering (ISESE'02).
28. Zubrow, D. (2004). Measuring Software Product Quality: the ISO 25000 Series and CMMI, Carnegie Mellon University - Software Engineering Institute.