Research Ideas and Outcomes 2: e8865
doi: 10.3897/rio.2.e8865

Grant Proposal

Software quality information needs

Daniel Graziotin [‡]

‡ University of Stuttgart, Stuttgart, Germany

Corresponding author: Daniel Graziotin (daniel.graziotin@gmail.com)

Reviewable v1

Received: 15 Apr 2016 | Published: 20 Apr 2016

Citation: Graziotin D (2016) Software quality information needs. Research Ideas and Outcomes 2: e8865. doi: 10.3897/rio.2.e8865

Abstract

Modern software methodologies, e.g., agile and continuous integration and deployment, rely on frequent feedback loops to respond to sudden changes. Improvements to feedback, development activities, and quality rely on information needs. We found a lack of understanding on the information needs of software engineers when the software quality is the main concern. I propose the term "software quality information needs" to refer to a software engineer’s eagerness to define, locate, obtain, and use information to satisfy a conscious or unconscious concern towards software quality. I want to understand what information about software quality does a developer need while performing code changes or designing new parts of a system. For understanding the broad research question, I propose three sub-questions, namely (1) How can we conceptualize information needs when dealing with software quality? (2) What information is needed when dealing with software quality? and (3) How can software quality information needs be detected unobtrusively through behavioral patterns? I propose a 24 months plan for a research fellowship comprising of three empirical studies, which will involve 120 students in computer science and the companies Daimler, Porsche, and Bosch. The studies will make use of multidisciplinary research in software engineering and behavioral science. The theoretical results of the execution of this proposal are the software quality information needs theory and instances, which will add to the body of knowledge in software engineering. The practical implications might be large. Providing a software developer with the right kind of information about the current state of and the effect of changes on software quality can prevent catastrophic software failures and avoid opening up security holes.

Keywords

information needs, software quality, process theory, affective reactions, behavioral patterns

State of the art

Software quality is a complex, multifaceted concept that is difficult to define because it depends on the specifics of organizations, projects, and people (Kitchenham and Pfleeger 1996). Yet, given the increasing pervasiveness of software in daily aspects of life and mission-critical tasks, we have to be concerned about the quality decay of software (Kitchenham and Pfleeger 1996). Regardless of how we define the quality of a software development process or product, we must measure the quality to determine if a system meets the defined quality goals (Deissenboeck et al. 2008). Hence, research in software engineering began to consider quality control programs.

Quality control emerged from the manufacturing area. Quality control means inspecting products during and after production to detect problems and ensure a high level of quality. Feigenbaum (1956) extended quality control to total quality control and later total quality management, which includes the idea of continuously improving the quality of processes and products. This has also been applied to software development. A common approach for quality control in software development has been statistical quality control. Because of the inherent differences between manufacturing and development processes, however, these approaches are not accurate in software development. Therefore, they have not been widely adopted in practice (Weller 2000). Among the possible reasons for the failure to adopt quality control approaches may be the complexity of the multifaceted construct of software quality, the heterogeneity of the information arising from the organizations, projects, and people (Kitchenham and Pfleeger 1996), and especially the speed and quality of the subsequent feedback.

Kamei et al. (2013) proposed just-in-time quality assurance for software development. They predicted defect-prone changes at code check-in time to reduce effort for inspecting changes. While this brings feedback very close to the developer and the corresponding task, it focuses on the narrow aspect of defect proneness. There is still the need to ensure a proper feedback to the developers.

Deissenboeck et al. (2008) emphasized the importance of feedback in quality control loops. They formulated the ideas of quality control and continuous improvement in software as continuous software quality control. The process is a loop of quality controls between quality goals, quality measurement, and feedback to the development teams. The feedback should happen frequently and as early as possible to cope with quality decay. The importance of feedback was highlighted by agile software methodologies, which are an established modern way to develop software. The ability to respond to sudden changes from the environment is a pillar of the foundations of agile software development, and the suggested way to responding to the changes is feedback loops (Williams and Cockburn

2003). These feedback loops have historically been intended as rapid feedback from and to the customer. Furthermore, the more recent agile methods of continuous integration and deployment provide a developer an immediate information if they broke the build or the software could not be deployed. Few research activities have considered feedback and information needs within the software development team.

The concept of information needs was coined in the areas of of library information systems and documentation. Taylor (1962) conceptualized the process of question asking in information retrieval, which includes the need for information. Information needs are divided into four parts: (1) actual, but unexpressed, need for information (visceral need), (2) the conscious within-brain description of the need (conscious need), (3) the formal statement of the question (formalized need), and (4) the question as presented to the information system (the compromised need). The conceptualization is not recent, and the aim of Taylor was to cope with question asking, not information needs. While the specifics of software quality information needs require a new conceptualization, Taylor's output will be helpful to frame the research of this proposal.

Wilson (1994) showed in his review that there has been extensive theory of information needs for information seekers, and that theory needs to be specialized for the domains in which the information is needed. Of interest for the present fellowship project proposal is that a need for information is preceded by psychological constructs of satisfaction (Wilson 1994), which can rather be re-conceptualized by affective terms together with evaluative judgments of the current information state. Several research domains have developed their own concepts for information needs and performed studies and experiments (Wilson 1994). Software engineering (or any related discipline) has yet to produce theoretical foundations for information needs in any context, software quality included.

The only mentioning of information needs theory in software engineering has been by Jedlitschka et al. (2014). However, they have applied the theory to the software manager's information need for technology selection. Few research activities in software engineering have dealt with the topic of information needs but no theory or theoretical framing was adopted.

Ko et al. (2007) argued that despite some research on information needs has been conducted over the years, we still know little about what information the developers look for, why they look for it, and what might prevent them to find it. For filling the gap, the authors conducted a two-month field study of co-located software developers at Microsoft. From the logged observations, they derived 21 general information needs divided into the following categories: writing code, submitting a change, triaging bugs, reproducing a failure, understanding execution behavior, reasoning about design, and maintaining awareness. The authors called for research on the automation of information acquisition. Several of the elicited questions are related to software quality, e.g., *why was this code implemented this way?*, *did I make any mistakes in my new code?*, and *is the problem worth fixing?*.

Fritz and Murphy (2010) argued that information needs carry some of the most tedious tasks for software developers. The authors interviewed eleven professional developers and identified 78 questions that developers often ask but are often unanswered as the information needs are unsupported. The questions have been divided into the broader categories of people specific, changes to the code, work item progress, broken builds, test cases, references on the web, and other. Among those questions, several are related to software quality, e.g, *what is the evolution of the code?* and *why were these changes introduced?* Fritz and Murphy (2010) do not explicitly discuss the information needs concept but investigate questions developers ask. A full quality information needs theory is missing.

The state of the art analysis has shown that there is a lack of understanding on the information needs of software engineers when the software quality is the main concern. Therefore, for this proposal, I build upon Taylor (1962) work to propose the term **software quality information needs** to refer to a software engineer's eagerness to define, locate, obtain, and use information to satisfy a conscious or unconscious concern towards software quality. I will discuss my prior related work in the context of the research questions.

Preliminary work

The present fellowship proposal is a standalone project. The proposal is self-contained. However, this proposal is framed in the context of a DFG grant proposal by Wagner (2015) on continuous and focused developer feedback on software quality.

While I have never worked on the specific topic of information needs and software quality, I consider my previous research experience to be much related to both the topics and the required theoretical and methodological underpinnings. For my PhD thesis work, I conducted much multidisciplinary research in software engineering with psychology. I have proposed the term *psychoempirical software engineering* and guidelines to conduct studies in empirical software engineering with psychology theory and measurements (Graziotin et al. 2015b). I possess expertise in bridging theory from industrial-organizational psychology and organizational theory to software engineering and to conduct research in empirical software engineering with psychology, both quantitatively and qualitatively. I would like to exploit my acquired expertise in this area for answering the research question.

For example, the first study (Graziotin et al. 2014) was among the first to understand the linkage between affect of software developers and their performance, and to advocate the use of psychology in software engineering. It reports an experiment with 42 computer scientists, who performed two tasks for assessing their problem solving performance according to their pre-existing affect. The affects of the developers were measured using a validated measurement instrument. The problem-solving performance was assessed using tests and measurements which are validated in the psychology fields. The study was published in a psychology venue (editor, reviewers) to test my capability to defend the acquired knowledge from the different domain of psychology. The article was of impact, as it is one of the most discussed articles in the world according to Altmetric.com (top 5% worldwide). Also, the implications for management styles suggest that fostering long-term positive affect in the software company will foster the problem solving- performance of developers.

The next study (Graziotin et al. 2015a) was the first to assess the correlation of the real-time affect triggered by a software development task and the productivity of the software developers. The affects of the developers were measured using a validated measurement instrument. The results have shown that the valence (the perceived attractiveness) of the task is a good driver of development productivity, almost as much as the dominance (feeling to dominate the current stimulus) of the task. The study was also among the first to employ linear mixed effects models as regression tools in the domain of software engineering. The results have implications for management styles, who should assign developers to their preferred tasks and support them to foster valence while working on the task.

The third study (Graziotin et al. 2015c) I would like to mention shows that I can excel in conducting qualitative studies and theory building studies. The study was an interpretive inquiry towards the construction of a novel explanatory theory of the impact of affects on development performance. The theory is novel in both fields of psychology and software engineering. The theory explains how events generate affective reactions, thus driving the focus on the current code and its progression towards goal settings (and reaching). The study introduces the novel concept of attractors, which are special affective states that dominate the conscious experience of the developers, thus fostering (or hindering) their performance dramatically.

Research objectives

The following are the research objectives of the present proposal.

Create a concept for software quality information needs (including "feedback")

A solid theoretical basis to conceptualize software quality information needs is required to deeply understand the phenomena and their complex interrelatedness. The acquired understanding will have theoretical implications for software engineering, as it will spawn new research activities building upon the currently missing building blocks for software quality information needs. We need the theoretical understanding for the next objectives, as the knowledge will enable the operationalization of the concept into measurable metrics.

Operationalize the software quality information needs concept into measurable metrics

We will exploit the acquired theoretical basis to operationalize the concept into measurable metrics. The metrics will be useful for quantitative representations (thus, assessment) of the software quality information needs, towards the creation of useful software quality feedbacks that are presented in the right context and at the right moment.

Develop unobtrusive measurement techniques for software quality information needs

Once the concept of software quality information needs is operationalized into quantitative representation, we will need to develop effective measurement techniques. Software development is an intellectual work that is undermined by useless and disturbing interruptions. Therefore, we will develop measurement techniques that are as unobtrusive as possible.

Evaluate the usefulness of the obtained metrics and measurement techniques

Our final aim is to conduct research that is useful to the industry and the society. Therefore, the development of the measurement techniques for the software quality information needs has to be accompanied by a practitioner-based validation of their usefulness. Once a validation of the usefulness measurement techniques is reached, we will consider the project as a very successful one.

Methodology

My research approach is mostly of empirical stance, and I identify my work in the discipline of empirical software engineering. Therefore, my proposed work builds upon methods that gather and analyze empirical evidence.

First, I intend to acquire a general understanding of the phenomena under study, as the construct needs to be explored, together with its causes and effects. The entire process concerning *software quality information needs* should be understood and modeled. Second, I intend to gather instances of software quality information needs to be employed in the theory and for the foreseen tool to improve the software quality. Third, I want to investigate how the process underlying *software quality information needs* and its instances can be gathered via behavioral metrics. A mixed research approach will be adopted. The mixed method approach recognizes that all methods have their limitations, thus it encourage triangulation through multiple data collection and analysis (Creswell 2009). Mixed methods approaches inquiry by combining qualitative and quantitative forms of research in different degrees (Creswell 2009). The following descriptions of mixed-research methods assume and imply a full compliance to the Declaration of Helsinki of ethical principles regarding human experimentation (World Medical Association 2013) in every of its ethical steps, such as informed consent, dignity protection, and data anonymization.

Research questions

Agile development and continuous quality control encompass the concept of short feedback loops. These feedback loops should be provided as quickly as possible. However, they still have a (sometimes fixed) length of minutes, hours, or days. The research shall be conducted at the Institute of Software Technology at the University of Stuttgart, where they work on providing immediate feedback to developers, which is related to software quality. Concretely, the institute is working on tools and methods to enhance the current state of just-in-time (as the code is written) source code analysis results in the context of software quality.

I intend to work on the feedback itself for understanding what constitutes a useful feedback as an answer to software quality information needs. My aim requires a deep understanding of the software developers' information needs and their detection while the developers deal with software quality issue. Thus, this fellowship proposal comprises of the stages of (1) conceptualization of the construct *software quality information needs*, (2) unpacking of the constructs into detectable (i.e., measurable) metrics to detect the information needs of a developer and to satisfy the information needs, (3) development of measurement and detection techniques that are as little intrusive as possible, and (4) empirically evaluating the usefulness of the obtained measurements to represent an information need.

On the practical side, the results of the proposed research activities may help to reduce the failures of software that we experience on a daily basis, including security concerns. I will now describe a scenario that will help to envision the applicability of the results of this proposal.

> *Anne is a software developer who is working on a banking application. Anne uses her favorite integrated development environment (IDE) for the development process. The IDE has a plugin which is created as a result of the proposed research activities, and it is able to detect software quality information needs and provide context-based, relevant feedback. Anne is working at a system routine that makes use of an Internet connection. She is not satisfied with the quality of the source code. Suddenly, a message in her IDE is telling her that the routine she is working on might contain some quality issues. Anne is surprised but at the same time relieved that the system has detected her software quality information needs. One issue is that she is employing a design pattern that is not frequently employed in similar cases. She is offered to browse some StackOverflow.com related questions and answers regarding the design pattern. Another issue is that she created part of the procedure by cloning code from another part of the project. The tool offers her help to refactor the cloned code. For the first issue, Anne acquires valuable information related to her issue. She is then able to use that information to fix the issues with her routine. For the second issue, Anne successfully refactors the code.*

Answering the following research questions will enable the creation of tools to support the envisioned scenario.

How can we conceptualize information needs when dealing with software quality?

The state of the art analysis has shown that some questions regarding information needs and software quality have been identified. However, we do not know what constitutes an information need in the context of software quality activities. We need a solid theoretical basis about what information about quality developers need during their work. Together with my peers, I will build on the existing work on information needs in software engineering and other disciplines. The answer to this research questions will be enabled by developing a theory for quality information needs based on existing information needs theories, results from existing studies and our own studies, which are described in the next section.

What information is needed when dealing with software quality?

In addition to conceptualizing the construct of *software quality information needs*, and its antecedents and consequences, there is the need to gather and analyze a significant amount of instances of information needs. Software developers require several pieces of information when dealing with the quality of the software, like a customer requirement or who is working on a chunk of code being refactored. Some studies have been conducted for eliciting instances of information needs of software developers, e.g., (Fritz and Murphy 2010, Ko et al. 2007). However, the main focus of the previous research was not software quality, which is instead the case of this research questions and the fellowship proposal.

Understanding and generalizing the construct and the instances of *software quality information needs* will enable the creation of better tools for supporting the quality improvement programs. As our scenario indicates, the software development tools will greatly benefit by this study research. In our scenario, the tool was able to detect an information need, its antecedents, and its context. Answering the research question and the related sub-question will enable us to create context-aware functionalities for enhancing the software quality of source code and the resulting programs.

The originality of the research question and the related sub-question is very high. To my understanding, there have not been any studies about *software quality information needs*. Although few studies have gathered instances of information needs for software developers, e.g., (Fritz and Murphy 2010, Ko et al. 2007), and some of which have gathered data of interest to software quality, there are no studies known to me which have addressed software quality as the main focus of study.

How can software quality information needs be detected unobtrusively through behavioral patterns?

Once a meaningful theoretical conceptualization for the construct *software quality information needs* is reached, the next step will be to study mechanisms for ensuring their detection. As software development is intellectual and cognitive based (Lenberg et al. 2015), we have to limit the developers' context switches and their concentration. Therefore, I will concentrate on non-invasive detection systems.

Wilson (1994) proposed that a need for information is preceded by psychological constructs of satisfaction. Therefore, I can assume that an information need can be conceptualized as affective reactions together with evaluative judgments of the current information state. More specifically, I hypothesize that affective reactions (or emotions) to source code fragments are linked, thus representative, of *software quality information needs*. A trivial example would be the arise of frustration when attempting to enhance the quality of a routine function. Thus, many ways to detect information needs would arguably be achieved through psychology measurements of behavior and affect. The use of psychology for studying human aspects in software engineering has often been proposed in recent times by several authors, e.g., (Lenberg et al. 2015). I am considered one of the few researchers to conduct proper research in software engineering with psychology theory and measurement (see for example Graziotin et al. (2014), Graziotin et al. (2015c), Graziotin et al. (2015a), Graziotin et al. (2015b), Kuzmickaja et al. (2015)).

The ability to detect information needs unobtrusively through behavioral patterns and affective reactions will enable the creation of tools that are are valuable to developers and will not break the flow while they develop software. As our scenario indicates, the tool was able to detect the information need before Anne was realizing it. This was achieved by observing Anne's behavior while she was developing. Speculated examples may be changes in the typing patterns and repeated line highlights over the code. Answering the research question will also enable future avenues, e.g., a distributed communication of information needs among a software development team.

There has been a recent literature review of psychological studies in software engineering by Lenberg et al. (2015), which covered several areas including those in which I have published during my PhD. However, information needs was not found by the systematic literature review. This opens up original research avenues of high impact for software quality. Thus, the research question is very novel. Its impact is high as well, because achieving non-intrusiveness for detecting *software quality information needs* will enable a higher software quality while non hindering the software developers' productivity.

Work packages

The research question *How can we conceptualize information needs when dealing with software quality?* will be answered empirically in **work package 1** by an observational study to investigate the small-scale information needs during typical development tasks. The study, represented as the first box in Fig. 1, will have computer science students as participants, as it has been found that computer science students are representatives of professional developers (Salman et al. 2015). The study will contain interviews and observations of students developing software. The study will recruit about 120 participants, which will be students of software engineering at Stuttgart University. Observational studies are an umbrella term for different empirical strategies. I see phenomenological analysis as most suitable for answering the question. (Interpretive) Phenomenological research is an inquiry strategy where the researchers identify the essence of human experiences about a certain phenomenon (Creswell 2009). The phenomenon is described by the participants, and the researchers should set aside their own experiences when describing the phenomenon. Phenomenological research combines psychological, interpretative, and idiographic components (Gill 2014). The study will have developers working on a real-world software project or a software project required for a university course. Most of the work by students will happen in groups, as software is mainly developed in teams. The researchers will observe them while they work, and interview them during their work by asking them to think-aloud. A post-task interview will enable more insights on the initial findings.

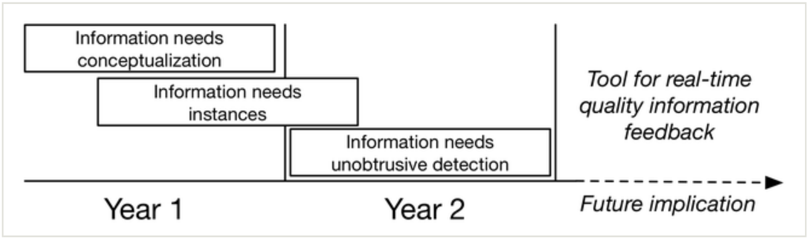
Information needs conceptualization
Information needs instances
Information needs unobtrusive detection
Tool for real-time quality information feedback
Year 1
Year 2
Future implication

Figure 1.

Timeline of the research activities and future implication.

The research sub-question *What information is needed when dealing with software quality?* will partially be answered in **work package 2** by the previously described study. In this case, however, we want to gather and prioritize instances of software quality information needs. The most suitable empirical strategy for this purpose is the survey. Surveys are a system for collecting information from or about people to describe, compare, or explain their knowledge, attitudes, and behavior (Fink 2003). Surveys enable having a breadth understanding of the factors under study. In particular, the questionnaires have been used to enable the collection and the analysis of data in a standardized manner which, when gathered from a representative sample of a defined population, allows the inference of results to the population (Rattray and Jones 2007). Surveys are administered to samples of a population to be queried for understanding a phenomenon. Thus, I propose to design a mostly open-ended survey to be distributed to the software industry. The proposed fellowship host has several connections with the industry, especially with the German automotive industry, which are deeply interested in software quality issues. We will target the software engineers of Daimler, Porsche, and Bosch. I intend to distribute the survey in order to obtain a broad view of instances of software quality information needs, and to prioritize them with respect to how they matter to practitioners. As illustrated by the second box of Fig. 1, this research will happen between year and 1 and 2 of the research activities of this fellowship.

The research question *How can software quality information needs be detected unobtrusively through behavioral patterns?* will be answered in **work package 3** through the conduction of a case study with interventions, which may be considered as in-field experiments or action research by some peers. Case studies, as defined by Thomas (2011), "are analyses of persons, events, decisions, periods, projects, policies, institutions, or other systems that are studied holistically by one or more methods. The case that is the subject of the inquiry will be an instance of a class of phenomena that provides an analytical frame--an object--within which the study is conducted and which the case illuminates and explicates" (p. 513). During a case study, researchers collect data using multiple data collection procedures over a sustained period of time (Creswell 2009). The case study will be proposed to the German multinational engineering and electronics company Robert Bosch GmbH. During the case study, I intend to propose and to test behavioral patterns, such as keystroke frequency, typos detection, voice analysis (if permitted by the company), and validated questionnaires on affect and job satisfaction, by gathering the pattern-related data and by interviewing the participants about the efficacy of the patterns. As illustrated by the third box of Fig. 1, this research will happen during the second year of the research activities of this fellowship, because it builds upon the results of the first two studies.

The three proposed studies are not of trivial nature, but their execution and analysis of the data will be unique for unlocking new avenues of research in software quality through the use of multidisciplinary research. The fellowship host's location and connections enable the engagement of different context (high number of students, Daimler, Porsche, and Bosch) that permit the gathering of unique and rich data. The theoretical implications of the execution of this proposal are the software quality information needs theory and instances, which will add to the body of knowledge in software engineering. As shown at the right side of Fig. 1, the practical implications of the execution of this research proposal will impact the software construction process itself, as they will allow the creation of context-aware tools for providing the right quality-related feedback at the right moment, in the right context.

Acknowledgements

I would like to thank Stefan Wagner for reviewing this proposal and for providing much useful feedback to it.

Hosting institution

Institute of Software Technology, University of Stuttgart, Germany

Ethics and security

The present fellowship proposal assumes and implies a full compliance to the Declaration of Helsinki of ethical principles regarding human experimentation (World Medical Association 2013) in every of its ethical steps, such as informed consent, dignity protection, and data anonymization.

Conflicts of interest

I am a subject editor at Research Ideas and Outcomes (RIO). I declare that I have no competing interests with respect to the present fellowship proposal.

References

- Creswell JW (2009) Research design: qualitative, quantitative, and mixed method approaches. 3. Sage Publications, 246 pp.
- Deissenboeck F, Juergens E, Hummel B, Wagner S, Parareda BMy, Pizka M (2008) Tool Support for Continuous Quality Control. IEEE Software 25 (5): 60-67. DOI: 10.1109/MS.2008.129
- Feigenbaum AV (1956) Total quality control. Harvard Business Review 43 (6): 93-101.
- Fink A (2003) The Survey Handbook. 2. Sage Publications Inc, 184 pp.
- Fritz T, Murphy G (2010) Using information fragments to answer the questions developers ask. 1. Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - ICSE '10. 175-184 pp. DOI: 10.1145/1806799.1806828
- Gill MJ (2014) The Possibilities of Phenomenology for Organizational Research. Organizational Research Methods 17 (2): 118-137. DOI: 10.1177/1094428113518348
- Graziotin D, Wang X, Abrahamsson P (2014) Happy software developers solve problems better: psychological measurements in empirical software engineering. PeerJ 2 (1): e289. DOI: 10.7717/peerj.289
- Graziotin D, Wang X, Abrahamsson P (2015a) Do feelings matter? On the correlation of affects and the self-assessed productivity in software engineering. Journal of Software: Evolution and Process 27 (7): 467-487. DOI: 10.1002/smr.1673
- Graziotin D, Wang X, Abrahamsson P (2015b) Understanding the affect of developers: theoretical background and guidelines for psychoempirical software engineering. Proceedings of the 7th International Workshop on Social Software Engineering - SSE 2015. 25-32 pp. DOI: 10.1145/2804381.2804386
- Graziotin D, Wang X, Abrahamsson P (2015c) How do you feel, developer? An explanatory theory of the impact of affects on programming performance. PeerJ Computer Science 1: e18. DOI: 10.7717/peerj-cs.18
- Jedlitschka A, Juristo N, Rombach D (2014) Reporting experiments to satisfy professionals' information needs. Empirical Software Engineering 19 (6): 1921-1955. DOI: 10.1007/s10664-013-9268-6
- Kamei Y, Shihab E, Adams B, Hassan A, Mockus A, Sinha A, Ubayashi N (2013) A large-scale empirical study of just-in-time quality assurance. IEEE Transactions on Software Engineering 39 (6): 757-773. DOI: 10.1109/TSE.2012.70
- Kitchenham B, Pfleeger SL (1996) Software quality: the elusive target [special issues section]. IEEE Software 13 (1): 12-21. DOI: 10.1109/52.476281
- Ko AJ, DeLine R, Venolia G (2007) Information Needs in Collocated Software Development Teams. 29th International Conference on Software Engineering (ICSE'07). 344-353 pp. DOI: 10.1109/ICSE.2007.45
- Kuzmickaja I, Wang X, Graziotin D, Dodero G, Abrahamsson P (2015) In Need of Creative Mobile Service Ideas? Forget Adults and Ask Young Children. SAGE Open 5 (3): 1-15. DOI: 10.1177/2158244015601719
- Lenberg P, Feldt R, Wallgren LG (2015) Behavioral software engineering: A definition and systematic literature review. Journal of Systems and Software 107: 15-37. DOI: 10.1016/j.jss.2015.04.084

- Rattray J, Jones MC (2007) Essential elements of questionnaire design and development. Journal of clinical nursing 16 (2): 234-243. DOI: 10.1111/j.1365-2702.2006.01573.x
- Salman I, Misirli AT, Juristo N (2015) Are Students Representatives of Professionals in Software Engineering Experiments? 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering Are. 666-676 pp. DOI: 10.1109/ICSE.2015.82
- Taylor RS (1962) The process of asking questions. American Documentation 13 (4): 391-396. DOI: 10.1002/asi.5090130405
- Thomas G (2011) A Typology for the Case Study in Social Science Following a Review of Definition, Discourse, and Structure. Qualitative Inquiry 17 (6): 511-521. DOI: 10.1177/1077800411409884
- Wagner S (2015) Continuous and Focused Developer Feedback on Software Quality (CoFoDeF). Research Ideas and Outcomes 1: e7576. DOI: 10.3897/rio.1.e7576
- Weller EP (2000) Practical applications of statistical process control [in software development projects]. IEEE Software 17 (3): 48-55. DOI: 10.1109/52.896249
- Williams L, Cockburn A (2003) Agile Software Development: It's about Feedback and Change. Computer, IEEE 36 (6): 39-43. DOI: 10.1109/MC.2003.1204373
- Wilson TD (1994) Information needs and uses: fifty years of progress? Aslib, 15-51 pp.
- World Medical Association (2013) World Medical Association Declaration of Helsinki-Ethical Principles for Medical Research Involving Human Subjects. JAMA 310 (20): 2191-2194. DOI: 10.1001/jama.2013.281053