New Perspectives on Software Quality

Ruth Breu, University of Innsbruck

Annie Kuntzmann-Combelles, inspearit

Michael Felderer, University of Innsbruck

0740-7459/14/$31.00 © 2014 IEEE

TODAY, INNOVATION IN software quality management is driven by the increasing role of quality within product innovation. Customers expect a certain level of maturity in software products and services, a level that turns software quality into a competitive advantage. A good example of this is e-shops, where customer-friendly interfaces as well as high performance for and correct handling of order processes significantly contribute to business success. Incidents caused by software failures such as a lack of availability of e-banking services for bank customers or a significant gap between predicted and actual delivery time of orders due to faulty logistics software must be dealt with as severe business risks.

Additionally, we now face a new generation of software systems, ranging from cloud and mobile services and cyberphysical systems to M2M (machine-to-machine systems). Emerging applications often carry a "smart" label (smart grids, smart medical devices, smart cities, and so on), and they have two aspects in common: a high potential for creating new markets or solutions to societal problems as well as the highest requirements for quality. Resilience, security, privacy, and safety are quality requirements that play a dominant role in contexts where the data of individuals and critical infrastructures are interconnected on a large scale.

A crucial implication emanating from these kinds of developments and a prerequisite for further considerations is that we must treat software quality management in a broader, more integrated context than we have in the past. Owing to the importance of software quality in product innovation, we understand the necessity for integrating software quality management with product management. The product owner role in Scrum is an important step in this direction. From the increased importance of software quality attributes such as security and resilience, we can deduce the necessity for integrating software quality management with IT management and systems operation. Secure infrastructures, in particular, are the result of a seamless process that integrates compliance considerations at the management level, design or purchase of secure software services, and configuration management and monitoring at runtime. Additional efforts are required for cyberphysical systems, where quality also encompasses hardware aspects. We propose the term "quality engineering" to stress this end-to-end aspect of software quality management (see Figure 1).

Grand Challenges to Quality Engineering

Taking the view of quality engineering as our baseline, we have deduced three major challenges for quality management of the emerging generation of software systems.

Interconnected Services

IT systems increasingly consist of fragmented services orchestrated in a decentralized way. Workflows typically comprise mobile services, cloud services, and sensor services. In such a context, quality management is not only required to cope with technical, multiplatform aspects but also with questions of trust and assessment of external services. Moreover, the openness of systems makes quality management indispensable for considering security at all levels of abstraction.

Systems Evolution

Future IT systems will be more evolutionary and more adaptive than

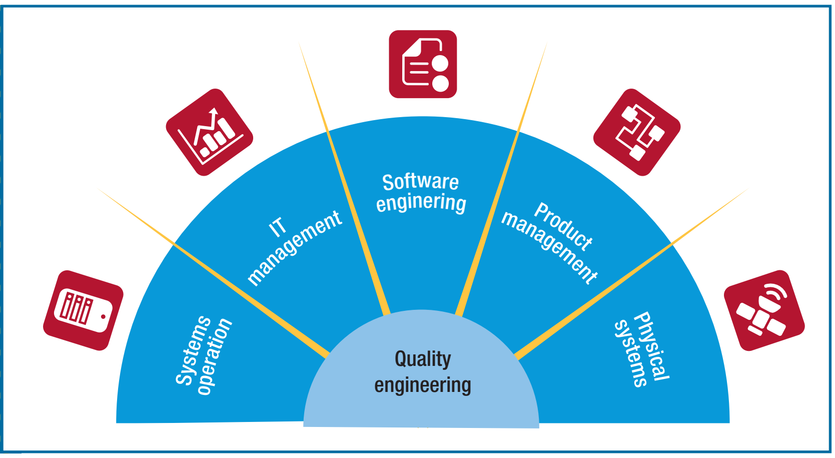

FIGURE 1. A diagram of quality engineering—end-to-end software quality.

ever before. Quality management in such a context requires an effective steering of quality management processes, powerful version management of all kinds of artifacts, and comprehension of interrelationships through effective traceability support. Automation is another crucial factor for increased efficiency. To meet the challenges of quality management in highly dynamic environments, we don't only need to improve automation of individual tasks (such as testing or verification), but we also need to improve the goal-oriented orchestration of automated, semiautomated, and manual quality management activities.

Stakeholder Collaboration

Although the global software engineering community has in recent years addressed aspects such as geographically distributed teams and outsourcing scenarios, additional challenges have emerged through increased collaboration of software engineers with executive and administrative roles. Crucial prerequisites for successful collaboration are user-centric processes and environments fitted to each stakeholder's specific tasks. This requires appropriate concepts and methods for information aggregation and distribution—knowledge to complete a task must be consistently presented within the domain concepts of the respective stakeholder (for example, at a business or technical level). In addition, it's crucial to motivate people to collaborate; this concerns, for example, support for and rewarding of documentation activities in various ways.

Future Directions for Software Quality Research and Development

Based on these general considerations about the future challenges in software quality management, we focus here on crucial areas for future research and development and some of the current approaches we deem to be groundbreaking. Because we take the end-to-end quality engineering concept as the baseline, the ultimate goal we are addressing in our statement is the management of software quality attributes from a business-oriented point of view and concepts, methods, and tools for the continuous fulfilment of these quality attributes within the productive use and development of IT services (see Figure 2). Crucial innovation includes

- *knowledge management*, which comprises aggregation, distribution, visualization of data, and information and knowledge to support collaborating stakeholders in fulfilling their quality-related tasks and decisions;
- *automation*, which comprises automated generation of artifacts and continuous efficient execution of automated and semiautomated tasks within software quality management;
- *data analysis*, which comprises the application of advanced data analysis techniques to evaluate current quality status, to predict future statuses, and to steer subsequent quality management tasks; and
- *collaborative processes*, which comprise goal-oriented orchestration of automated, semiautomated, and manual quality management tasks in an environment of stakeholders collaborating across organizational levels and boundaries.

Knowledge Management

The knowledge base of quality engineering encompasses a wide range of structured and unstructured information, comprising code repositories, requirements specifications, legal regulations, test reports, tickets in project management systems, and system configurations, just to name a few. One of the major goals of knowledge management in this context is to provide each stakeholder with information to fulfill his or her tasks in the best possible way, to help them detect risks at an early stage, and to let them interact properly with other stakeholders. This gives us a clear set of requirements for the knowledge base:

- Information must be of sufficient quality—the knowledge base should strive for properties such as consistency and actuality.
- Information must be interlinked to support stakeholder interaction and data analysis.
- Information must be presented within stakeholders' domain concepts; this requires both information transformation (for example, aggregation) and appropriate visualizations.
- In contexts where stakeholders from various organizations (and organizational levels) interact with each other, the knowledge base must provide sophisticated mechanisms to ensure confidentiality and integrity.
- Owing to the size and complexity of such a knowledge base, advanced retrieve and search functions are essential.

A visionary approach for knowledge management addressing several of these aspects is the concept of view-based software engineering, in which stakeholder-centric views are generated from a central knowledge base.[1] The Open Services for Lifecycle Collaboration community has proposed a different, more pragmatic direction, defining standards and lightweight interfaces to support tool chains in software engineering (http://open-services.net).

Information traceability isn't a new topic in software quality management, although it is one of increased importance in upcoming settings. In the future, it will become more important to supplement manual maintenance of information links by automated and semiautomated tool support.[2] This will increase the quality and number of links between arbitrary software engineering artifacts and is the basis for new, stakeholder-specific visualizations and data analysis techniques required for managing evolving and complex IT systems.

Requirements management is an example of a subdiscipline with paramount necessity for improved knowledge management. Not only do service-centric systems and cloud services imply myriad requirements and configurable features in manifold versions and variants, but flexible offers to customers also require close collaboration among account managers, requirements engineers, test managers, and software architects. Although several approaches address these types of challenges,[3,4] we must also consider that defining requirements is a creative process in which stakeholders and analysts work together to create ideas expressed as system requirements. The piloting of techniques and tools developed to stimulate the creativity during this collaborative activity indicates that increased novelty doesn't necessarily come at the expense of decreased quality and usefulness of the requirements.[5]

Data consistency and actuality are important yet nebulous goals because a knowledge base for quality engineering encompasses a wide range of information from manifold sources, both in structured and unstructured form. Therefore, we don't only need formal but also semiformal techniques. For example, well-orchestrated manual, semiautomated, and automated tasks help

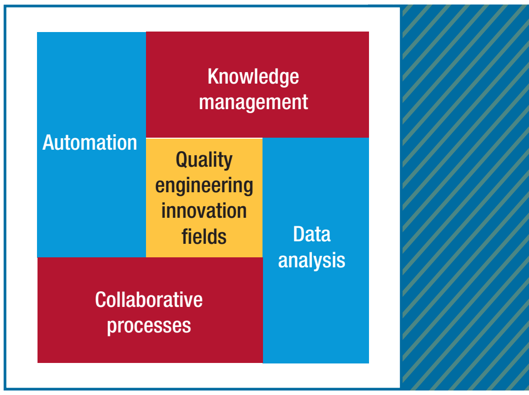

FIGURE 2. Quality engineering innovation fields.

keep enterprise architecture models in sync with the actual IT infrastructure.[6] As another example, consider that early detection of conflicts between security and safety requirements addresses challenges to certification of cyberphysical systems.

Automation

A cornerstone of the continuous quality management of evolutionary and adaptive IT systems is automation. Generative techniques producing executable artifacts (such as system code and tests) have attained an attractive level of productivity in recent years. Models play a major role within automation because they can help represent systems at different levels of abstraction. In this way, models enable business-oriented systems representation, the management of varying service technologies, or specification across system and platform boundaries at a uniform level of abstraction.

Modern modeling workbenches provide flexible support for domain-specific languages. On the basis of metamodel definitions, they accommodate powerful model engineering support ranging from editors and model transformation to code generation. XText (www.eclipse.org/Xtext) and MPS (www.jetbrains.com/mps) are examples of such generic modeling workbenches supporting textual domain-specific modeling languages. Others have demonstrated the potential and practical relevance of a modeling workbench instantiation for the embedded software domain, supporting model-based abstractions on C code, automated testing, requirements tracing, and formal program verification.[7] Another example is Txture (http://txture.org), which integrates use of textual modeling and graphical visualizations in the IT architecture documentation domain.

Model-based testing approaches automate the generation of test inputs, drivers, stubs, and oracles on the basis of models of a system or its environment. The benefits of model-based testing include early and explicit specification and verification of system behavior, transparent test design and documentation, scalable and effective test automation, and support for managing changes. Therefore, when regression testing is performed to provide confidence that changes don't harm existing behavior, the benefits of model-based testing will usually far outweigh its costs of model creation and maintenance. Recently, search-based techniques such as evolutionary and genetic algorithms[8] have been especially successful in generating test artifacts.[9]

Automation beyond executable artifacts—supporting artifact traceability, requirements management, or data analysis—still has a huge potential that is waiting to be unlocked.

Data Analysis

Continuous build and deployment processes, ticket-based project management tools, and automated test environments are valuable data sources for advanced data analysis techniques. Data analysis can provide us powerful support for decision making by helping us evaluate the current quality status, predict future statuses, and recommend subsequent quality management tasks.

Manifold metrics have been defined to evaluate a system's quality status. In practice, it's neither possible nor meaningful to characterize software products or projects by a single quality metric. Therefore, software dashboards, which integrate advanced metrics that are often visualized as graphs, optimally support decision making. In addition, software dashboards contribute to increased transparency in the software development process as well as additional team awareness. So it's especially important to provide stakeholder-specific visualizations and search functions to optimally support, for example, project managers, product managers, software architects, or testers so they can perform their tasks well.

It's neither possible nor meaningful to characterize software products or projects by a single quality metric.

Risk-based testing, which uses risk information to optimize all phases of the test process, has a high practical relevance as far as coping with limited testing resources is concerned. A core activity in every risk-based testing process is risk assessment. In current practice, risk assessment is mainly performed manually and in an ad hoc manner. This makes risk assessment and therefore the overall risk-based testing process expensive, time-consuming, and nondeterministic regarding human decisions. There is still big potential to develop a more methodical and scalable approach by integrating automatically determined metrics into the risk assessment process.[10]

To predict future quality status of software, various bug prediction models have also been proposed. The most important benchmarked and compared bug prediction categories are process metrics, previous defects, source code metrics, entropy of changes, and churn and entropy of source code metrics.[11] Manual data validation for obtaining useful prediction results is one recommendation for misclassification on bug prediction.[12] Another promising application area of bug prediction is the use of predictions in the context of risk assessment.

With the increased availability of a huge amount of static and dynamic data from software engineering and IT management processes, new opportunities for improved decision support based on powerful prediction techniques and user-centric visualizations are likely to arise.

Collaborative Processes

It's quite clear that the new kinds of fragmented, adaptive, interacting systems of services pose formidable challenges to the overall quality engineering process. First, the future quality engineering process encompasses new interfaces, stakeholders, and quality management tasks, all of which must be orchestrated. Among these are the increased importance of quality management of nonfunctional requirements such as security and performance, the integration of hardware- and software-related quality management, and quality management of external (cloud) services. There's a proposal for a general approach for quality management of cyberphysical systems integrating manifold hardware and software models available,[13] as well as a thorough discussion about the new role of testing in the era of the cloud.[14]

Other challenges arise from the need to integrate strictly structured processes in IT management and agile processes within software engineering. A change of paradigm from project-centric processes to more general change-driven processes for quality engineering integrating manual and automated tasks has been proposed.[15] We are still in need of a foundational approach for steering processes by results of data analytics (for example, predicting quality attributes or recommending quality related activities such as testing).

Although we as guest editors are coming from the modeling community, we have not ranked model-based approaches as a future focal area in this issue per se. Rather, we deem model engineering as a core enabler to achieve goals such as automation, stakeholder-centric views, and prediction. However, we also see the necessity to depart from "raw" model-based approaches and focus on quality goals instead. On the one hand, this means, for instance, that model engineering (model transformation, versioning, and the like) is performed in the background while stakeholders are provided with environments best adapted to their tasks. On the other hand, the end-to-end quality engineering approach requires integration techniques that work on structured (model-based) and unstructured data, for instance, to improve data retrieval and data analysis.

Continuous delivery is mandatory for business agility.

In This Issue

This special issue, owing to its fundamental software quality focus, comprises a collection of diverse articles. They address the challenges and directions for software quality research as we have discussed in this introduction.

In "On the Accuracy of Automated GUI Testing for Embedded Systems," Ying-Dar Lin, Edward T.-H. Chu, Shang-Che Yu, and Yuan-Cheng Lai introduce a method for bypassing the uncertainty of runtime execution environments to guarantee that GUI operations are reproduced at the device under test. Smart phones are used as devices under test in the article, and the experimental results achieved for an Android platform are very encouraging. Their approach is therefore a step forward in automating test activities for interconnected services.

In "Data Protection in Healthcare Social Networks," Jingquan Li discusses healthcare social networking sites—websites that provide users with tools and services to easily establish contact with each other around shared problems and to utilize the "wisdom of crowds" to attack diseases. The article presents two case studies showing that these networks must be secure and avoid unauthorized use of disclosure of patient data to implement collaborative processes and to adequately share knowledge to support collaborating stakeholders.

In "Economic Governance of Software Delivery," Murray Cantor and Walker Royce tackle the dynamic agile transition observed in many organizations. Fast-evolving, competitive systems and environments force companies, large to small, to consider an in-depth transformation of both mindsets and activities to stay alive. Continuous delivery is mandatory for business agility. Measurement to steer evolution and support continuous improvement must be revisited. This article offers an interesting new data analytics approach based on Bayesian reasoning to improve predictability for economic governance.

Luís da Silva Azevedo, David Parker, Martin Walker, Yiannis Papadopoulos, and Rui Esteves Araújo's article "Assisted Assignment of Automotive Safety Requirements" and its consequences are essential for evolution and knowledge management in the automotive industry. Owing to the high number of LOC and developments such as driverless vehicles, safety requirements become more and more important, as well as more difficult to manage. Because Automotive Safety Integrity Levels

ABOUT THE AUTHORS

RUTH BREU is a full professor at the University of Innsbruck. Her research interests include software engineering, information security, enterprise architecture management, requirements engineering, and model engineering. Breu received a habilitation (postdoctoral qualification) in computer science from Technische Universität München. Contact her at ruth.breu@uibk.ac.at.

ANNIE KUNTZMANN-COMBELLES is a founder and CEO at inspearit. Her research interests include agile and lean management, process management, and metrics. Kuntzmann-Combelles received an MS in aerospace engineering from Sup'Aéro. She's an associate editor of *IEEE Software*. Contact her at annie.combelles@inspearit.com.

MICHAEL FELDERER is a research associate within the Quality Engineering research group at the Institute of Computer Science at the University of Innsbruck. His research interests include software quality in general, software testing, requirements engineering, and empirical software engineering. Felderer received a PhD in computer science from the University of Innsbruck. Contact him at michael.felderer@uibk.ac.at.

(ASILs) are difficult to perform manually, the authors present a new technique that automates the allocation and decomposition of ASILs to support the system and software engineering life cycle as well as system evolution and knowledge management in this context.

In "Decision-Centric Architecture Reviews," Uwe van Heesch, Veli-Pekka Eloranta, Paris Avgeriou, Kai Koskimies, and Neil Harrison focus on assessing system architecture quality, which is important to foster stakeholders' expectations, especially for interconnected services where architecture plays a key role. Their presented approach to architecture evaluation uses architecture decisions as first-class entities. The approach uncovers and evaluates the rationale behind the most important architecture decisions, considering the entire context, in which the decisions were made. It fosters stakeholder collaboration, supports knowledge management, and provides valuable input for data analysis.

Together, the articles collected in this special issue on software quality contribute to all the innovation areas we've mentioned and point us in the direction of solving the grand challenges of quality engineering.

References

1. C. Atkinson, "Orthographic Software Modelling: A Novel Approach to View-Based Software Engineering," *Modelling Foundations and Applications*, LNCS 6138, Springer, 2010, p. 1.
2. J. Cleland-Huang, O. Gotel, and A. Zisman, *Software and Systems Traceability*, Springer, 2012.
3. K. Pohl, G. Böckle, and F. van der Linden, "Software Product Line Engineering: Foundations, Principles and Techniques," Springer, 2005.
4. S. Lohmann et al., "Semantifying Requirements Engineering—the Softwiki Approach," *Proc. 4th Int'l Conf. Semantic Technologies* (I-SEMANTICS 08), J. UCS series, 2008, pp. 182–185.
5. K. Zachos and N. Maiden, "Inventing Requirements from Software: An Empirical Investigation with Web Services," *Proc. 16th IEEE Int'l Requirements Engineering Conf.*, IEEE, 2008, pp. 145–154.
6. M. Farwick et al., "Automation Processes for Enterprise Architecture Management," *Proc. 15th IEEE Int'l Enterprise Distributed Object Computing Conf. Workshops* (EDOCW 11), IEEE, 2011, pp. 340–349.
7. M. Voelter et al., "mbeddr: Instantiating a Language Workbench in the Embedded Software Domain," *Automated Software Eng.*, vol. 20, no. 3, 2012, pp 339–390.
8. M. Harman, S.A. Mansouri, and Y. Zhang, "Search-Based Software Engineering: Trends, Techniques and Applications," *ACM Computing Surveys*, vol. 45, no. 1, 2012, article 11.
9. R. Matinnejad et al., "Automated Model-in-the-Loop Testing of Continuous Controllers Using Search," *Search Based Software Engineering*, vol. 8084, Springer, 2013, pp. 141–157.
10. M. Felderer et al., "Integrating Manual and Automatic Risk Assessment for Risk-Based Testing," *Software Quality: Process Automation in Software Development*, Springer, 2013, pp. 159–180.
11. M. D'Ambros, M. Lanza, and R. Robbes, "Evaluating Defect Prediction Approaches: A Benchmark and An Extensive Comparison," *Empirical Software Eng.*, vol. 17, nos. 4–5, Springer, 2012, pp. 531–577.
12. K. Herzig, S. Just, and A. Zeller, "It's Not a Bug, It's a Feature: How Misclassification Impacts Bug Prediction," *Proc. Int'l Conf. Software Eng.*, IEEE, 2013, pp. 392–401.
13. J. Sztipanovits, "Cyber Physical Systems—Convergence of Physical and Information Sciences," *it-Information Technology Methoden und innovative Anwendungen der Informatik und Informationstechnik*, vol. 54, no. 6, 2012, pp. 257–265.
14. H.M. Sneed, "Testing Web Services in the Cloud," *Software Quality: Increasing Value in Software and Systems Development*, Springer, 2013, pp. 70–88.
15. R. Breu et al., "Living Models-Ten Principles for Change-Driven Software Engineering," *Int'l J. Software and Informatics*; doi:10.1109/CISIS.2010.73.

See www.computer.org/software-multimedia for multimedia content related to this article.