Software Qualities and their Dependencies Report on two editions of the workshop

Séverine Sentilles[1], Barry Boehm[2], Catia Trubiani[3], Xavier Franch[4], and Anne Koziole[5]

[1]Mälardalen University, Västerås (Sweden), severine.sentilles@mdh.se

[2]USC, Los Angeles, CA (USA), boehm@usc.edu

[3]Gran Sasso Science Institute, L'Aquila (Italy), catia.trubiani@gssi.it

[4]Universitat Politècnica de Catalunya, Barcelona, Catalonia (Spain), franch@essi.upc.edu

[5] Karlsruhe Institute of Technology, Karlsruhe (Germany), koziolek@kit.edu

DOI: 10.1145/3375572.3375581

ABSTRACT

http://doi.acm.org/10.1145/3375572.3375581

New trends in software engineering recently emerged to cope with even more complex systems which in turns highlight problems software shortfalls and defects. The SQUADE (Software QUAlities and their DEpendencies) workshop focuses on increasing the understanding of the nature of Software Qualities (SQs), i.e., non-functional properties or extra-functional requirements (e.g., reliability, security, maintainability, etc.), and their interrelationships with the aim of bringing them into practice of software engineering. The topic is highly relevant due to the current trend of designing and developing software-intensive systems with larger complexity, increased autonomy, higher speed of changes, and growing need for interoperability within systems of systems. Unfortunately, this new trend comes with more software shortfalls and defects, which are widely and publicly spread. The primary goal of the workshop is to bring together researchers and practitioners to build more solid foundations when dealing with software qualities.

During the first edition of the workshop, we identified a number of SQ topics which need further research, namely, the inadequacy of existing SQ standards, the potential benefit of machine-learning to the field and the latent needs for regulatory practices. This year we are pleased to have received many submissions covering several SQ topics such as security and privacy, accessibility, modifiability, interoperability, reliability, development vs. runtime metrics. The growing interest in these topics is promising, and may foster many other future editions of the workshop, thus to build a large community available to join forces in this domain.

1. INTRODUCTION

The increasing prevalence of software in today's society calls for the ability to develop high quality software at a faster pace. Most of the existing solutions in software engineering have been quite successful at taming software complexity as far as the sole functionality is concerned, whereas quality has been less considered so far. Software quality specifies how well systems and software perform their functionality and encompasses properties such costs, performance, reliability, availability, security, etc.

The importance and timeliness of SQUADE is highlighted by current and future trends toward more software-intensive systems. These systems are characterized by larger complexity, autonomy, speed of change, and need for interoperability within systems of systems, simultaneously demand of higher levels of safety, security, scalability, adaptability, multi-cultural usability, speed of development and evolution, and affordability. These systems are found in Internet of Things (IoT), Cyber-Physical Systems (CPS), massive-data analytics, self-driving cars, dynamic global supply chains, e-Health, and Industry 4.0, among many others new emerging domains.

An example of the current chaos among SQ practices, definitions, standards and relationships is the current standard in the area, i.e., ISO/IEC 25010. For example, it defines reliability as the *"degree to which a system, product, or component performs specified functions under specified conditions for a specified period of time"*. As a standard, this statement is supposed to hold for any definition of *"specified functions"* and *"specified conditions"*. However, for agile methods for example, *"specified functions and conditions"* are often sunny-day stories and use cases, and a system will then be judged to be ISO/IEC-reliable if it satisfies only the specified sunny-day conditions (but nonetheless fails on the rainy-day conditions). Further, the definition can be considered restrictive, in fact it focuses on performing functions only.

Another source of chaos is the diversity of SQ definitions. As a simple example we can look at the definition of "*Resilience*", as defined in Wikipedia. The definitions differ between and within the domains of Ecology, Energy Development, Engineering and Construction, Network, Organizational, Psychological, and Soil. The differences are indeed non-trivial as they include ten different definitions of the system's post-resilience state.

In this chaotic context, the SQUADE workshop aims at bringing together researchers and practitioners working with different aspects of software qualities and their dependencies. The goal is to discuss the current state-of-the-art and state-of-practice, existing solutions, and to identify open challenges that must be addressed to improve the different software qualities. In this note, we summarize the discussions and findings of the two editions of the workshop. The first one was held on May 27th, 2018 in Gothenburg, Sweden in conjunction with International Conference on Software Engineering (ICSE), 2018, and the second one on August 26, 2019 in Tallin, Estonia together with the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), 2019.

The rest of this note is organized as follows: Section 2 gives a brief overview of the structure of the workshop, whereas Section 3 highlights key research issues which have been discussed by the

participants of the workshop during the breakout sessions. Section 4 concludes the report by highlighting the key take-away from the two first editions of the SQUADE workshop.

2. WORKSHOP FORMAT

The workshop's main objective is to foster discussions and exchanges of ideas between participants coming from different fields. As a result, SQUADE gives prominence to breakout sessions and mingle opportunities and is typically organized with:

- an **opening session and keynotes**,
- a set of **flash presentation sessions**, during which the authors of accepted papers give a brief overview of their contribution in 10-15 minutes;
- a set of **breakout sessions**, i.e., the participants brainstorm on topics identified during the presentation sessions.

In the first edition of the workshop, Harold "Bud" Lawson, the 2000 Computer Pioneer Award for his invention of the pointer variable, gave an invited talk on an overview of an Automatic Train Control System with a special focus on its related software qualities. An interesting particularity of this system is that it has been in use since the 70s. Besides, seven papers were presented on topics including user's satisfaction, usability, code quality, software maturity, timing properties, maintainability, static analysis, validation/verification and testing.

In the second edition of the workshop six papers were selected to foster discussions at the workshop. The discussed topics included security and privacy, accessibility, modifiability, interoperability, reliability, development vs runtime data. An increasing need of sharing experiences emerged, in fact there was an initiative for building Free/Open Source Software (FOSS) along with their qualities. Chatbot platforms have been proposed as test-beds to quantify software qualities and evaluating their interactions. Predictions and preliminary estimations have been recognized of key relevance since they may anticipate flaws that are very difficult and expensive to be fixed in later stages of the software development process.

3. IDENTIFIED RESEARCH ISSUES

During the two editions of the workshop, a number of topics related to software qualities and their dependencies which need further research were identified, namely, the inadequacy of existing SQ standards, the potential benefit of machine-learning to the field and the latent needs for regulatory practices. This year we receive many submissions covering several SQ topics such as security and privacy, accessibility, modifiability, interoperability, reliability, development vs runtime data. A summary of the discussions on these issues from the two editions of the workshop are described hereafter.

3.1 Inadequacy of definitions and standards

When dealing with software qualities in software engineering, the first question is whether the reference material is actually sufficient. Most of the definitions and standards do have implicit assumptions, which obfuscate their understanding and limit their applicability. As a result, people often do not understand SQs in the same way, which introduce unnecessary chaos and additional complexity in the development process.

In spite of this, the ISO standards have become the de-facto standards (e.g., lifecycle management, the software quality [1]). The workshop participants, however, unanimously questioned their usefulness to improve software quality as they provide ambiguous software quality definitions and do not support well enough the dependencies between qualities. For example, in ISO 25010 [1], testability is a sub-category of maintainability. However testability influences many other properties such as reliability, security. It is thus questionable whether it should be a sub-category of maintainability, a category in its own right, or a sub-category of another property. Besides this structural aspects, another problem with static standards is that they become rapidly obsolete and do not keep up with the latest trends in software engineering, such as new upcoming laws and regulations (e.g., GDPR [2]), or the emergence of new properties (e.g., sustainability [3]).

The workshop participants identified the following research directions to improve the work on software qualities:

- Use of (supervised and unsupervised) machine-learning techniques to learn from a system under analysis the most relevant software qualities and their dependencies;
- Use of empirical studies to investigate the opinions of different stakeholders (e.g., end-users, developers, project managers) when evaluating the software qualities;
- Revise the definitions and standards to take into consideration the context of software qualities and their dependencies, thus to adapt these definitions when needed.

3.2 Limited regulation practices

As stated in [4, 5], "*IT governance is typically the weakest aspect of corporate governance*", which means little focus is put on satisfying software qualities unless it is required. Increased regulatory and governance practices may be a solution to enforce companies to pay more attention on software qualities. Some governments and international agencies have already started to envisage such options as visible, for instance through the creation of the new EU-law on General Data Protection Regulation [2].

This lead the workshop participants to identify the following open research questions which would be worth investigating further:

- How the new regulations can be integrated in current development practices?
- What would be the long-term effects of these regulations? How to maintain the developed software?

3.3 Quality-unaware technological evolution

Technology is running ahead providing solutions that no one asked for initially, and omitting quality in the process. For example, with the emergence of multicore ECUs, it becomes difficult to have a look at the software as an isolated black-box. This leads to trade-off between predictability, accuracy, and costs. On the other hand, systems such as smartphones work astonishingly well, even though it can be argue that some SQ-related features should be better (e.g., battery consumption).

Besides, another problem is that code is often preferred over design and analysis which can be seen as opposite to the practices in system engineering which conversely favor models. This might be due to the fact that code is cheap to change and education in software engineering often first teaches programming, even before modelling. However, it is generally acknowledged that only developing software through manual coding (not to say hacking when facing increasing timing constraints) introduces lots of issues in software development [6].

Lastly, wrong decisions are being taken which negatively affect software qualities and their perceptions. Often, decisions are

purely taken according to the estimated costs with little regard for software qualities. In this context, the following solutions were suggested by the participants:

- Make software qualities a tangible (visible and payable) property for end-users;
- Establish a holistic approach integrating software governance, process, people, and software artifacts;
- Report on best practices and on negative outcomes (through a failure repository for instance);
- Propose novel and better solutions for immediate quality feedback during software development.

4. CONCLUSION

An important conclusion of the two first editions of the workshop is that software qualities and their dependencies are not sufficiently established in today's software systems. As the complexity of systems is staggeringly raising, the lack of support for evaluating SQs and their dependencies is certainly going to become an more important issue to be further investigated. In particular, most of the work on SQs and dependencies is currently pair-wise only. However, SQs are more like a n-dimension matrix of dependencies (whether positive or negative) that are also affected by other characteristics, such as the software architecture, the implementation code, the deployment infrastructure, etc. This implies that a strong correlation between properties is not enough to make any conclusion. It is equally important to state when such a dependency is valid and thus find ways to identify how to define the context in which a dependency is valid.

Acknowledgements

We would like to thank all the workshop participants who provided really interested contributions and actively enable lively discussions. First edition of the workshop has been attended by: Christopher Ehmke, Lori Flynn, Mattias Gålnander, Wahab Hamou-Lhadj, Mahshid Helali Moghadam, Harold Lawson, Jinhua Liu, Dor Maayan, Saad Mubeen, Shola Oyedeji, Efi Papatheocharous, Birgit Penzenstadler, Ralf Reussner, Kamonphop Srisopha, Miroslaw Staron, Sichao Wen, Florian Wessling, Uwe Zdun. Second edition of the workshop has been attended by: Daniela Micucci, Jinjing Zhao, Matthias Miller, Menglong Yang, Michael Felderer, Saurabh Srivastava, Aytaj Aghabayli, Julian Harty, Faiz Ali Shah, Xiang Li, Mathias Ellmann.

5. REFERENCES

[1] ISO/IEC, "ISO/IEC 25010 System and Software Quality Models," tech. rep., 2010.

[2] "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)." https://eur-lex.europa.eu/eli/reg/2016/679/oj, April 2017. (Accessed on 10/14/2019).

[3] C. Calero, M. F. Bertoa, and M. Á. Moraga, "Sustainability and Quality: Icing on the Cake," in *International Workshop on Requirements Engineering for Sustainable Systems (RE4SuSy)*, 2013.

[4] S. P.-J. Wu, D. W. Straub, and T.-P. Liang, "How Information Technology Governance Mechanisms and Strategic Alignment Influence Organizational Performance: Insights From A Matched Survey Of Business And It Managers," *MIS Quarterly*, vol. 39, no. 2, pp. 497–518, 2015.

[5] M. Ghobakhloo, "The future of manufacturing industry: a strategic roadmap toward Industry 4.0," *Journal of Manufacturing Technology Management*, vol. 29, no. 6, pp. 910–936, 2018.

[6] C. C. Mann, "Why Software Is So Bad," *Technology Review*, vol. 105, no. 6, pp. 33–38, 2002.