MeasureSoftGram: A Future Vision of Software Product Quality

Hilmer Rodrigues Neri
UnB/FGA and COPPE/UFRJ
Brazil
hilmer@unb.br

Guilherme Horta Travassos
COPPE/UFRJ
Brazil
ght@cos.ufrj.br

ABSTRACT·

Background: Software product quality assurance affects the acceptance of releases. The one dimensional observational perspective of current software product quality (SPQ) models constrains their use in continuous software engineering environments. Aims: To investigate multidimensional relationships between software product characteristics and build an evidence-based infrastructure to observe SPQ continuously. Method: To mine and manipulate datasets regarding software development and use. Next, to perform multidimensional analytical SPQ interpretations to observe quality. Results: There is empirical evidence on the multidimensionality linkage of quality characteristics throughout the software life cycle. Conclusions: The one-dimensional quality perspective is not enough to observe the SPQ in continuous environments. Alternative mathematical abstractions should be investigated.

CCS CONCEPTS

Software and its engineering → Software creation and management; Quality Assurance

KEYWORDS

Software Product Quality, Software Analytics, Release acceptance, Continuous Software Engineering, Continuous Experimentation, Empirical Software Engineering.

ACM Reference format:

Neri H.R; and Travassos G.H; 2018. *In Proceedings of ACM International Symposium on Empirical Software Engineering and Measurement, Oulu, Finland, October 11-12, 2018 (ESEM'18), four pages.*
DOI: https://doi.org/10.1145/3239235.3267438

1 INTRODUCTION

Software engineers have studied software product quality (SPQ) for nearly four decades. The reference models usually categorize SPQ factors as a hierarchy of quality characteristics and sub-characteristics. Also, some studies are concerned with their associated measurements and relationships between them [1-2-3-4]. Empirical evidence emphasizes the need to observe the relationships among the quality characteristics when analyzing SPQ [3-7-8]. In open source development, other quality characteristics and sub-characteristics were defined [2]. ISO 9126 and its update ISO 25000 have been the most studied SPQ models in the literature [27]. A recent study demonstrated the relationships between ISO 25010 quality characteristics [4].

It is possible to observe in the previous standard SPQ models [1-6] a lack of experimental evidence regarding the relationships among factors, concentrating most of their findings in opinions obtained in surveys of the industrial sector. Besides, they make difficult to aggregate studies due to semantic differences and the use of a one-dimensional perspective (observing a quality characteristic per time without considering side effects or influences of other characteristics) to observe software quality, constraining its observation. Also, empirical studies on measurement and software quality measures commonly report a lack of rigor in the measurement process and definition of measures [23]; problems in scales between measures [19]; the difficulty of establishing reference values for software measures [24]; and finally, the characteristics of the context that make difficult, or even impossible, the comparison of measures between different products or projects [25]. All of these may jeopardize to observe the quality of software products in operation; the software behavior in use, and consequently, the organization's business strategy. In some sense, these issues have been addressed in recently proposed ISO-based SPQ models such as SQuale, Quamoco and particularly in QATCH. [28-29-30]. These models provide a good advance in the operationalization of the missing connections between abstract descriptions of software quality characteristics and specific software analysis approaches. It turns out such models also observe SPQ using a one-dimensional perspective, focusing on internal quality. We argue that relationships between SPQ characteristics must be explicitly exposed and analyzed in order to effectively capture the whole spectrum of quality.

Also, software engineers' perception on how to develop software changed. It has been increasingly perceived more holistically. According to this vision, activities and tasks from the development process are organized and executed to establish a continuous cadence of workflows. It contributes to characterize the "continuity" in software engineering environments. Usually, such environments offer an entwined set of development and deployment activities and tasks organized and executed continuously and no longer offered as a set of phases, activities, and tasks, organized and executed discretely [5]. Therefore, they

ESEM'18, October 11-12, 2018, Oulu, Finland FIN

ACM ISBN 978-1-4503-5823-1/18/10. $15.00
DOI: https://doi.org/10.1145/3239235.3267438

allow the continuous collection of development, experimentation, and software usage data.

Due to strong engineering principles in such development environments, continuous and systematic SPQ assurance is mandatory in this new development reality. However, the limitations of current SPQ models restrict their direct use in such environments. For instance, the one-dimensional perspective and a *posteriori* measurement do not support a holistic quality observation *on-the-fly*. This problem directly affects the decision making on the releases acceptance regarding what should be deployed and delivered for use at a particular level of quality.

The acceptance of releases represents a challenge for software organizations. Notably, in our case, it is even more critical when dealing with public software development contracts in Brazil. Usually, the releases are accepted *ad-hoc*, without the capacity of observing characteristics that can affect internal, external, and usage product quality altogether. Therefore, we believe a multidimensional and continuous perspective to address and observe quality can represent a vision of the future and contribute to reducing the acceptance risks in our organizations.

We realized it because the multiple relations between quality characteristics and sub-characteristics reveal multiple interactions and side effects among measures. The side effects of these relations will only be observable when the characteristics and sub-characteristics are analyzed together; hence evidence on their adherence to the phenomena and relationships must be explicitly known and used in the analysis. To guide this ongoing research, we formulate the following question:

How to perform multidimensional SPQ observations in continuous engineering environments to improve the decision making on release acceptance?

As far as we could observe, multiple relationships among quality characteristics and sub-characteristics must be captured and represented in the n-dimensional space, considering the differences in the measurement scales and their need for aggregation and comparison. Therefore, SPQ characteristics and their sub-characteristics should be represented by a multidimensional and interrelated structure in the space of quality. A mathematical concept allowing a multidimensional representation in the space is the tensor [18]. In this sense, the notion of tensors can be understood as a generalization of vectors and scalars concepts. Comparable tensors in space can represent different variables and scales, including their mutual influence. In our case, each of the quality characteristics and its sub-characteristics can be represented through tensors, called from now on tensors of quality.

The following sections depict our vision on how to use a multidimensional perspective to observe SPQ in contemporary software engineering environments continuously. Besides this introduction, Section 2 describes basic concepts regarding our research. Next, our proposal is presented, including the different pieces of technology that could make it available. Section 4 presents some conclusions for this vision paper.

2 BACKGROUND

2.1 Continuous Software Engineering

Continuous engineering principles and practices have been progressively adopted in the industry [9-5-10]. It recognizes that the software development process activities must be managed holistically. Continuous engineering principles mean that activities carried out by expert teams (with clear divisions of roles and following a rigid and defined workflow) make it challenging to adapt software changes besides to prevent the autonomy of the development teams. It compromises the cadence of software delivery and damages organizations' business as it increases waste of effort.

Besides the continuous development process cadence, one of the most notable changes is concerned with the high level of automation in delivering and deployment software releases. Such automated practices and technologies have been allowing different software organizations to take advantage of better balancing the time-to-delivery pressures, by instrumenting the DevOps pipeline [11] and thus breaking the limitation of automation technologies as observed in the 70´s [6].

2.2 Continuous Experimentation

Over the last ten years, experimental strategies have been proposed in continuous environments. Continuous experimentation (CE) allows to base decisions on releases acceptance of hypothesis testing. According to statistical testing results, a particular version of a product is chosen and deployed [12-26]. However, it usually focuses on usability and usage characteristics of web applications using simple experimental arrangements, such as A/B testing. In this type of experiment, half of the sample (users) is exposed to the treatment (new version), and later it is compared to the version in operation (control). At the moment of comparison, when the hypothesis test result is taken into consideration, the chosen version is deployed to the whole population of users. Typically, the selection and randomization of the sample are done from active user sessions, therefore during software operation.

Recent works proposed generic models for adopting CE practices. Fabijan *et al.* report a consolidated model acquired from over ten years working at Microsoft [14]. Fagerholm *et al.* propose a model based on observing CE use in startups [13]. Schermann *et al.* present an overview of CE techniques, which helps to understand this practice's state and suggest that organizations need foundational support to move from an experience-driven culture to experiment-driven culture in the decision-making on release acceptance [15].

2.3 Software Analytics

Mining techniques in software repositories are currently remarkable as well as the high availability of projects' data. Automation of mining, mathematical and statistical analysis have boosted research in the area of software analytics. Although there is no standard definition of software analytics, there is a common understanding that it refers to the process by which a computational solution examines data extracted from development

and operational environments, making use of mathematical methods with the purpose of finding patterns and relations to obtain insights that drive the technical-managerial decision-making [16-17].

2.4 Basics of Tensors and their Analysis

An array of components represent a tensor. Components are coordinates functions of space. Therefore, a vector is less general than a tensor. The order of a tensor defines its dimension. Therefore, a vector is a particular case of a tensor of order 0 [18]. One of the interests of multilinear algebra is the n-dimensional vector space with p-linear applications. It is similar on what we can observe with the quality characteristics and their relationships. There are multilinear algebra foundations, such as canonical bilinear isomorphism, commutative and associative properties that can support numerical transformations in the scalars (in this research, the quality measures). The tensor product enables us to do mathematical operations of sum and product of tensors and it is thus represented: $U \otimes V$, let U and V are vector spaces of m e n dimensions. The linear case can be extended to p-linear, generalizing its application. Then, there is a canonical bilinear application expressed by: $\phi : V_s^r \otimes V_{s'}^{r'} \rightarrow V_{s+s'}^{r+r'}$. This application is called tensor multiplication. It can be used, for instance, to combine different quality characteristics. Another important operation with tensors is the contraction used to reduce its dimensionality. Through contraction we can reduce a tensor to a linear application. It generalizes the canonical bilinear form $U \otimes V^* \rightarrow \mathbb{R}$ being thus defined: $c_j^i : V_s^r \rightarrow V_{s-1}^{r-1}$.

By doing so, the mathematical properties of the tensor product are supposed to enable the treatment of software quality in multidimensional and multivariate vector spaces homogenously.

3 OUR PROPOSAL: The MeasureSoftGram

One of the challenges regarding a multidimensional observation of quality regards the different perspectives and measurement scales used to represent the quality characteristics, sub-characteristics, and their relationships. Scale transformations and quality characteristics aggregation become a complicated problem whether following the one-dimensional perspective precluded by SPQ models. Therefore, we believe that tensors algebraic application should mitigate the problems related to the measurement scales that usually invalidate experimental studies and making the aggregations of behaviors hard [19-20]. So, of tensors of quality will model the quality measures after statistical analyzes and treatments.

In our case, the challenge is to observe the quality of releases and decide about their deployment and delivering in continuous environments. Due to the dynamicity of the development process, the expectation of quality evolves in each interaction and use of the released product, turning the continuous observation of quality an explicit requirement for software engineers.

In this development and releasing scenario, each release (R_i) has a defined duration (Δ_{Ti}). Throughout R_i planning activity, the product quality measurement objectives should be planned based on different stakeholders' perspectives [21-22]. It is assumed that R_i is developed according to a continuous delivering flow and after Δ_{Ti} a decision-making on the release acceptance happens. So, one or more quality characteristics (and sub-characteristics) will represent the quality measurement objectives. The quality measures, mined and extracted in static or dynamic way, in turn, will support the observation of quality characteristics including their side effects (captured through the relationships among them). Thus, a product quality baseline must be defined at each $\Delta_{Ti(start)}$ and at $\Delta_{Ti(finish)}$. At R_i acceptance activity, the previously defined baseline will be compared with the realized measures and, next, evolved and becomes the baseline for the next release. Each baseline is called a product quality configuration - SPQC_i. Therefore, for all R_i there will be a $\text{SPQC}_{i\,(planned)}$ at $\Delta_{Ti(start)}$ and $\text{SPQC}_{i\,(realized)}$ at $\Delta_{Ti(finish)}$. Although it sounds common place, these quality settings will be spatial modeled as tensors, representing n-dimensional evidence-based quality characteristics arrays, as defined in some certain SPQ models, as well as its ${T_{QCi}}_s^r$ relationships. Thus, we understand that the multiplication of these tensors of quality provides us with a vector measure, on a ratio scale, allowing the comparison between different product quality configurations, as following:

$$\text{SPQC}_i = {T_{QCi}}_s^r \otimes {T_{QCi'}}_{s'}^{r'} \otimes {T_{QCi''}}_{s'}^{r'}, \ \ldots, \otimes {T_{QCn}}_{s_n}^{r_n} \qquad (1)$$

An overview of our proposal is presented in Fig. 1. The next paragraphs depict it. Due to the sake of space, we will only describe the main components of our approach, marked in the figure as the steps (I) and (VI).

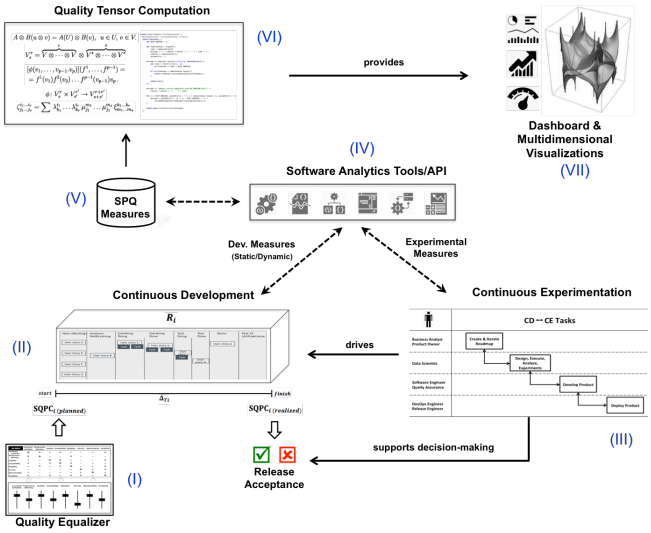

Figure 1: Continuous and Multidimensional Quality Observation in MeasureSoftGram.

The Quality Equalizer (QE) – step I - supports the tuning of a software product quality configuration in the MeasureSoftGram, as shown in Fig. 2. The stakeholders involved in the release acceptance decision making can use the QE to define their quality expectations for R_i, which will be represented by

the $\text{SPQC}_{i\,(planned)}$. When R_i is evaluated at $\Delta_{Ti(finish)}$, $\text{SPQC}_{i\,(realized)}$ will be measured in the developed product version, and compared to $\text{SPQC}_{i\,(planned)}$. Consequently, the assessment of SPQC_i will take place through evidence obtained from data, which will guide the decision-making regarding R_i acceptance. At the starting point (R_1) or when no data is yet available, the effects of ISO 25010 relationships, as identified by Lavazza and Morasca [25], are going to be used to build ${T_{QC1}}_s^r$ w. Therefore, each ${T_{QCi}}_s^r$ represents a multidimensional set of measures of a given quality characteristic (and sub-characteristics) with their relationships. The relationships between the characteristics are captured by the tensor indices, which by definition are the covariant and contravariant tensors.

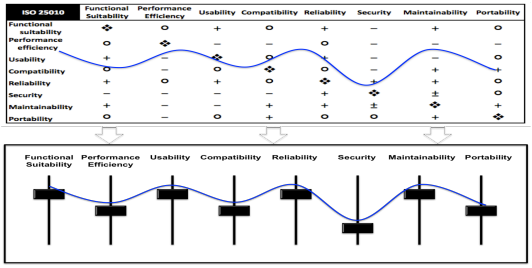

Figure 2: MeasureSoftGram Quality Equalizer.

The quality measures mined and extracted from the continuous environments (II and III) will be statistically treated and processed by software analytics tools (IV), then providing measures to build the ${T_{QCi}}_s^r$ and support further comparison between them, systematizing the decision making support.

These data will be stored in a repository of quality measures (V), so preserving static and dynamic measures; plans and experiments results, beyond the results of decisions. The DevOps practices can ensure the complete and automated reproduction of operational environments, analysis, and decisions that were mutually agreed.

We believe all of these will allow us to look at and compare continuously SPQC_i using a multidimensional quality perspective when performing SPQ.

4 CONCLUSIONS

This paper described our vision of the future of supporting the continuous and multidimensional observation of software product quality in continuous software environments. It has been called MeasureSoftGram. Our motivation to deal with this problem resides in the context of Brazilian Public Software Development Contracting, which operates a considerable amount of public resources in different software projects annually. To improve the decision making on the acceptance of software product releases is vital to protect the public resources and guarantee that high-quality software products (under the perspective of stakeholders) are available to the Brazilian society.

We are currently instrumenting a computational infrastructure in the Experimental Software Engineering Lab at COPPE/UFRJ to support all of the MeasureSoftGram features. It will allow evaluating the feasibility of our visionary proposal. Later it is intended to observe its use in different development contexts, such as startups and innovative software companies, besides Brazilian public organizations.

ACKNOWLEDGMENTS

UnB, CAPES, and MinC support this work. Prof Travassos is a CNPq researcher (grant 305929/2014-3). Thank designers Putu K., Abhishek R., Rafael G. M., Icon I., Hea P. L., to share images (icons) under Creative Commons license from Noun Project.

REFERENCES

[1] McCall, J.A. & Richards P.K. & Walters G.F, Factors in Software Quality, Volumes I, II, and III. US Rome Air Development Center Reports, US Department of Commerce, USA, 1977.
[2] Miguel J.P. & Mauricio D. & Rodrígues G., A review of software quality models for the evaluation of software products. IJSEA, 2014
[3] Henningsson K. & Wohlin C., Understanding the relations between software quality attributes - a survey approach. In Proceedings of 12th ICSQ, 2002
[4] Haoues M. & Sellami A. & Ben-Abdallah H. & Cheikhi L., A guideline for software architecture selection based on ISO 25010 quality-related characteristics, IJSAEM, 2017.
[5] Fitzgerald B. & K.J. Stol., Continuous software engineering: A roadmap and agenda, JSS, 2017.
[6] Boehm B. W., Characteristics of Software Quality, Volume 1, TRW Series, North-Holand, 1978.
[7] Aldaajeh S. & Asghar T. & Abed A.K. & Zakaullah M., Communing Different Views on Quality Attributes Relationships' Nature. EJSR, 2012.
[8] Svahnberg M. & Henningsson K., Consolidating different views of quality attribute relationships, WoSQ@ICSE, 2009.
[9] Dybå T. & Dingsøyr T., Agile Project Management: From Self-Managing Teams to Large-Scale Development, ICSE, 2015.
[10] Poppendieck M. & Poppendieck T., Implementing Lean Software Development: From Concept to Cash, (The Addison-Wesley Signature Series), 2006.
[11] Debois P., DevOps: A software revolution in the making, JITM, 2011.
[12] Kohavi R. & Longbotham R. & Walker T., Online Experiments: Practical Lessons, IEEE Computer, 2010.
[13] Fagerholm F. & Sanchez G. A. & Mäenpää H. & Münch, J, The RIGHT model for Continuous Experimentation. JSS, 2017.
[14] Fabijan A. & Dmitrie P. & Olsson H. & Bosch J., The Evolution of Continuous Experimentation in Software Product Development, ICSE, 2017.
[15] Schermann G. & Cito J. & Leitner P. & Zdun U. & Gall H. C. We're Doing It Live: A Multi-Method Empirical Study on Continuous Experimentation, IST, 2018.
[16] Buse R.P.L. & Zimmermann T., Analytics for software development, FoSER, 2010.
[17] Menzies T. & Zimmermann T. Software Analytics: So What?, IEEE Software, 2013.
[18] Lima E.L., Tensorial Calculus, Vol. 32 de Math Notes, IMPA, 1965
[19] Juristo N. and Moreno A.M., Basics of Software Engineering Experimentation, Springer Publishing, 2010.
[20] Wohlin C. & Runeson P. & Host M. & Ohlsson M.C., Regnell B. & Wessln A., Experimentation in Software Engineering, Springer Publishing, 2012.
[21] Buse R.P.L. & Zimmermann T., Information needs for software development analytics. ICSE, 2012.
[22] Fenton N. & Bieman J., Software Metrics-A Rigorous and Practical Approach, 3th, CRC Press, 2014.
[23] Kitchenham B., What's up with software metrics? – A preliminary mapping study, JSS, 2010.
[24] Lavazza L. & Morasca S., An Empirical Evaluation of Distribution-based Thresholds for Internal Software Measures, PROMISE, 2016.
[25] Dybå T. & Sjøberg D. & Cruzes D., What works for whom, where, when, and why? On the role of context in empirical software engineering, ESEM, 2012.
[26] Bosch J., Building products as innovation experiment systems, LNBIP, 2013
[27] Ouhbi, S. et al., Evaluating Software Product Quality: A Systematic Mapping Study, IWSM-MENSURA , 2014
[28] Mordal-Manet K. et al., The squale model – A Practice-Based Industrial Quality Model, ICSM, 2009
[29] Wagner S. et al., The quamoco product quality modeling and assessment approach, ICSE, 2012
[30] Siavvas M. G. et al., QATCH - An adaptive framework for software product quality assessment, ESWA Journal, 2017