Empirical Studies and Measurement of Software

Special track along with 16^{th} International Conference on Software Engineering Advances,
ICSEA 2021, October 4-6, 2021 in Barcelona, Spain

Luigi Lavazza
Dipartimento di Scienze Teoriche e Applicate
Università degli Studi dell'Insubria
Varese, Italy
email:luigi.lavazza@uninsubria.it

***Abstract*—This paper summarizes four presentations in the special track "Empirical Studies and Measurement of Software." The research work deals with the following key issues of this track:**

- **Adapting well-established measurement practices to evolving technologies and processes;**
- **Exploiting emerging technologies to improve the measurement process;**
- **Evaluating newly proposed code measures.**

This publication shows that the contributions in this track address research questions that are of high importance for industrial practice as well as current research.

Keywords–Cognitive complexity; software code measures; McCabe complexity; cyclomatic complexity; Halstead measures; static code measures

I. INTRODUCTION

Software measures are of paramount importance for the software development industry. Quite interestingly, there are a few measure that were proposed long ago and still play a fundamental role in software processes. An example of such measures are Function Points, which were proposed by Albrecht in the 70's [1] and are still the most widely used measure on which effort estimation is based. Alternatives to Function Points, like COSMIC Function Points [2], have been in use for a couple of decades and testify about the importance of functional size measures, in general. Another set of measures that have been defined long ago and are still in use include Halstead metrics [3] and several maintainability indices [4]–[6].

Although the aforementioned measures are widely used, and partly *because* they are quite popular, there is the need to adapt them to new types of software and new types of development processes. When considering functional size measures, the application of traditional Function Points or COSMIC Function Points in agile processes is a challenge, because of several reasons: requirements are usually non entirely known and detailed at the beginning of the development process, different "agile-specific" ways of carrying out sizing and effort estimation were proposed (story points, planning poker, etc.), people involve in agile development are not very well disposed towards documents and measurement, ...

Software product organization is continuously evolving: consider that software applications, once monolithic, became client-server, then multi-tier, then service-based, and nowadays are often organized into microservices. Therefore, it is hardly surprising that software measures must be continuously adapted to capture aspects that previously did not exist.

Although a huge number of measures were proposed for software through years, new measures are continuously proposed. Adopting those measures implies acquiring new tools, as well as the knowledge needed to understand and use the measures. In addition, the repositories of historical data have to be updated, and processes have to be modified to exploit the knowledge conveyed by new metrics. Quite clearly, all this involve a huge organizational effort. Thus, software developers should spend time and money to adopt new measures only if it has been demonstrated that the new measures do actually convey some knowledge that "traditional" measure are not able to provide.

The papers submitted to the "Empirical Studies and Measurement of Software" special track at ICSEA 2021 deal with the aforementioned hot topics. A brief description of the papers is given in the following section.

II. SUBMISSIONS

The "Cognitive Complexity" measure was introduced with the aim of improving the capabilities of McCabe complexity [7] in indicating code that is difficult to understand and maintain. "Cognitive Complexity" accounts for a few characteristics of source code, including the number of decision points (e.g., `if`, `for`, `while` and `switch` statements) and the level of nesting of control statements. When the "Cognitive Complexity" measure was proposed, no evaluations were published concerning its relationship with traditional code measures. In his paper [8], Lavazza has reported about an empirical study aiming at evaluating the correlation between "Cognitive Complexity" and several traditional measures, including those addressing the same characteristics of code taken into account by "Cognitive Complexity." To this end, Lavazza measured a few open source projects' code, obtaining the measures of 3,610 methods. He then performed statistical analysis using both correlation tests (namely, Kendall's and Spearman's rank correlation coefficients) and regression analysis. He found strong correlations to McCabe's complexity and slightly less strong correlations to several other code measures. Several regression models of "Cognitive Complexity" as a function of traditional measures were found. Not surprisingly, one of the most accurate models involves McCabe's complexity, NLE (Nesting Level Else-If) and LLOC (the number of logical lines of code) as independent variables. Considering that the best models found are quite accurate, Lavazza concludes that, at least for the considered software projects, "Cognitive Complexity" does not appear to convey additional information with respect to traditional measures.

Fehlmann and Kranich addressed the application of the COSMIC functional size measurement method in agile processes [9]. In agile software development, story points are used to indicate the effort needed to implement a user story, and to track the progress of a software product under development. One of their major limitations is that they do not allow predicting the number of sprints needed to create or modify a software product. Fehlmann and Kranich consider functional size measurement methods (namely, IFPUG or COSMIC methods) a more promising way to measure sprints. However, the functional size of the product is not simply determined by the total functionality implemented in sprints. Agile teams often re-work the same functionality, because new requirements concerning existing functionality must be implemented, and some already implemented functionality is removed. Moreover, activities like refactoring and testing require some effort, which can be measured in story points, but add no functionality. Fehlmann and Kranich investigate via a case study how effort and functional size growth depend on each other in sprints. Specifically, for each sprint, they measured the functional size of new developments, enhancements and rework. They conclude that the difference between size of the product and total size of all sprints, plus the amount of enhancement works, reflects the effort needed to find the correct requirements by the agile team. Functional sizing allows to better understand the percentage of effort that is needed for non-functional requirements, refactoring and testing and may vary strongly per sprint. Typically, "nonfunctional" effort accounts for more than half of the total effort; thus, the value of functional sizing for sprint planning is limited. However, for predicting the number of sprints needed to reach a minimum viable product, and for managing DevOps, functional sizing is without alternative.

Pedraza-Coello and Valdès-Souto address the application of the COSMIC measurement method [2] to software structured according to the Microservices Architectural Style [10]. Specifically, Pedraza-Coello and Valdès-Souto observe that highly coupled microservices lead to both design-time problems, because of high interdependence between development teams, and run-time problems involving latency and network traffic. Being able to measure the coupling between microservices in early phases of the software development life cycle could help the software architects make better design decisions. Therefore, they propose a way of measuring coupling between microservices, based on the COSMIC measurement method. Specifically, the proposed coupling measure is the count of the entries and exits data movements of the functional processes of a given micro-service that involve other micro-services. Pedraza-Coello and Valdès-Souto illustrate the application of the proposed measure in a simple software application.

The functional size measurement processes suffer from a few well-known drawbacks: one is that the measurement activities can be rather long and expensive, another one is that the measures are subjective, to some extent. Both these problems stem form the fact that functional size measurement is carried out by people, following measurement manuals that do not provide uniquely interpretable measurement rules. Therefore, it is hardly surprising that functional size measurement automation, from software specification documents, has been a research topic over the last several years. In industry, software requirements are often written in natural language. The recent advances in natural language processing (NLP) are enabling the automatic extraction of valuable information from text. NLP makes it possible to extract elements that support functional size measurement from software requirements specifications written in natural languages. Gèrançon et al. [11] propose a NLP-based tool that extracts "triplets" (composed of subject, predicate and object) from text describing use cases or user stories. The triplets belonging to the same functional process are then identified, thus making it possible to measure the size of the functional process in COSMIC Function Points, every triplet representing a data movement.

III. Conclusion

The research results that have been presented at the "Empirical Studies and Measurement of Software" special track at ICSEA 2021 address hot topic in the field of software measures and measurement. It is interesting to note that all the session's papers tend to confirm the utility and validity of well-consolidated software measures: it is thuis worth investing and researching in evolving, adapting and perfecting those measures in advanced environments.

Acknowledgment

My great thanks go to the organizers for the kind reception of this special track in the conference. This may include the considerable administrative effort. Special thanks to the many reviewers who gave the authors many valuable hints. Much thanks to authors for the very interesting contributions and the willingness to publish and present them.

References

[1] A. J. Albrecht, "Measuring application development productivity," in Proceedings of the joint SHARE/GUIDE/IBM application development symposium, vol. 10, 1979, pp. 83–92.

[2] Common Software Measurement International Consortium (COSMIC), "The COSMIC Functional Size Measurement Method - version 4.0.2 Measurement Manual (The COSMIC Implementation Guide for ISO/IEC 19761: 2017)," 2017.

[3] M. H. Halstead, Elements of software science. Elsevier North-Holland, 1977.

[4] T. Kuipers and J. Visser, "Maintainability index revisited–position paper," in Special session on system quality and maintainability (SQM 2007) of the 11th European conference on software maintenance and reengineering (CSMR 2007). Citeseer, 2007.

[5] P. Oman and J. Hagemeister, “Metrics for assessing a software system’s maintainability,” in Proceedings Conference on Software Maintenance 1992. IEEE Computer Society, 1992, pp. 337–338.

[6] I. Heitlager, T. Kuipers, and J. Visser, “A practical model for measuring maintainability,” in 6th international conference on the quality of information and communications technology (QUATIC 2007). IEEE, 2007, pp. 30–39.

[7] T. J. McCabe, “A complexity measure,” IEEE Transactions on software Engineering, no. 4, 1976, pp. 308–320.

[8] L. Lavazza, “An Empirical Study of the Correlation of Cognitive Complexity-related Code Measures,” in Proceedings of the 16th International Conference on Software Engineering Advances – ICSEA 2021, 2021.

[9] T. Fehlmann and E. Kranich, “Functional Size Measurement in Agile,” in Proceedings of the 16th International Conference on Software Engineering Advances – ICSEA 2021, 2021.

[10] R. Pedraza-Coello and F. Valdès-Souto, “Measuring Coupling in Microservices using COSMIC Measurement Method,” in Proceedings of the 16th International Conference on Software Engineering Advances – ICSEA 2021, 2021.

[11] B. Gèrançon, S. Trudel, R. Nkambou, and S. Robert, “Software Functional Size Automation from Requirements Written as Triplet Structure,” in Proceedings of the 16th International Conference on Software Engineering Advances – ICSEA 2021, 2021.