Managing Corrective Actions to Closure in Open Source Software Test Process

Tamer Abdou
CSE Department
Concordia University
Montréal, Québec, Canada
t_moh@encs.concordia.ca

Peter Grogono
CSE Department
Concordia University
Montréal, Québec, Canada
grogono@encs.concordia.ca

Pankaj Kamthan
CSE Department
Concordia University
Montréal, Québec, Canada
kamthan@encs.concordia.ca

***Abstract*—In assessing test process maturity, one of the goals is to manage disciplinary issues. Managing corrective actions to closure is known to aid software quality assurance, in general, and testing process activities, in particular. In this paper, a framework for software testing assessment, namely OSS-TPA, that aims to evaluate corrective actions in OSS test process, is proposed. The OSS-TPA framework is based on earlier studies and relies on a conceptual model for test process activities in OSS development. Using success factors in OSS development, the relationship between the maturity of managing corrective actions and the adoption of OSS is investigated.**

***Index Terms*—Open Source Software; Software Engineering; Software Quality; Software Testing; Test Process Improvement.**

I. Introduction

In the past couple of decades, there has been a notable growth in the adoption of open source software (OSS), both by organizations and by people. The increasing commitment to OSS places ever more moral and ethical responsibility on the developers to produce better software. This, in turn, impacts the OSS development process, and calls for attention to OSS quality, in general, and OSS testing, in particular.

In this paper, the interest is in the improvement of OSS test process [1]. Indeed, process improvement (along with automation and standardization) is regarded as one of the major research directions in software testing [2]. To that end, this paper proposes an OSS test process assessment framework, henceforth abbreviated as OSS-TPA, that provides guidelines, procedures, and metrics with the aim of evaluating OSS projects.

In recent years, a number of maturity models have been proposed for evaluating OSS projects. However, these models do not focus on the underlying OSS development process, and do not adequately address issues related to testing or the maturity of the underlying testing process. This motivates the need for evaluating the OSS testing process systematically, and the OSS-TPA framework is a step in that direction. OSS-TPA is based on the Test Maturity Model Integration (TMM*i*) [3]. Furthermore, OSS-TPA relies on a conceptual framework that identifies OSS test process activities, such as Test Design and Implementation, Test Execution, and Test Incident Reporting, and aligns these activities with the ISO/IEC Standard for test process [4].

The rest of the paper is organized as follows. Section 2 analyzes existing approaches for assessing and measuring OSS projects, as well as examines related studies on the assessment of OSS test process. Section 3 provides a description of the OSS-TPA framework and its parts. Section 4 suggests avenues for future research. Finally, Section 5 provides concluding remarks.

II. Background and Related Work

The approaches for investigating the OSS test process can be classified into two categories, namely assessment and measurement [5]. These are discussed in some detail in the next two sections.

A. Assessment Approaches

An assessment approach is concerned with qualitative evaluation. In this approach, reasoning or subjective judgment is taken to conclude whether the OSS or one of its software components meet specified requirements.

A number of assessment models have been introduced over the years to provide the basis for evaluating the test process of software projects, as summarized in Table I [6]. TMM*i* is a successor of these initiatives. It provides guidelines and a reference model for test process improvement, and has proven useful in practice [3] [7] [8]. The TMM*i* reference model has sixteen process areas that include practices, ranging from general to specific, related to test process improvement. Furthermore, each process area is subdivided into a number of goals to be achieved in order to reach a specific level of maturity.

B. Measurement Approaches

A measurement approach is concerned with quantitative evaluation. In this approach, direct measures are recorded and compared to pre-established values to decide whether the OSS or one of its software components meet numerical thresholds.

The metrics for test process allow managers to track, understand, and control (and thereby improve) testing. For example, the number of test cases, defect density, and other similar metrics, provide an insight into different aspects of a test process [9] [10].

Features	TMM	TIM	TPI	TMM*i*
Model Type	Maturity	Maturity	Maturity	Maturity
Year of Development	1996	1996	1997	2008
Approach	Theoretical	Practical	Practical	Theoretical
Number of Levels	5	5	14	5
Number of Key Process Areas	13	5	20	16
Assessment Type	Questionnaire	Questionnaire	Checklist	N/A
Assessment Foundation	CMM, ISO, SPICE	Practical Experience	Practical Experience	CMMI
Information about Model	Articles, Thesis, Books	Articles	Articles, Books	Articles, Books

TABLE I: An Overview of the Main Features of Existing Test-Process Improvement Models

In recent years, a number of maturity models have been proposed for evaluating OSS projects, including OpenBQR [11], OpenBRR [12], SQO-OSS [13], and FOCSE [14]. They aim to help prospective adopters understand the features of an OSS, and to assess the advantages and drawbacks of its selection and use [15] [1].

However, these models are rather limited in their consideration of process maturity, in general, and test process maturity, in particular. For example, out of twenty-eight evaluation criteria in OpenBRR, only two criteria (namely, the average volume of the mailing list in the last six months and the number of unique contributors in the last six months) are relevant to process maturity [12]. In some maturity models, criteria for test process maturity (such as, the criterion of the availability of testing and benchmark reports in OpenBQR [16]) are mentioned, but not considered in any detail. Finally, these maturity models lack a standard basis [12], and the process of capturing data that these models are derived from is usually subjective [17]. The purpose of OSS-TPA is to overcome some of these drawbacks.

III. The OSS-TPA Framework

The OSS-TPA framework consists of four modules, namely Quality Model, Data Collection, Data Analysis, and Data Interpretation, as shown in Figure 1.

A. *OSS-TPA Quality Model*

The OSS-TPA quality model is based on two complementary approaches to satisfy the definition of test process evaluation [18], and thereby evaluate the OSS test process.

The first approach is the use of the Goal-Question-Metric (GQM) framework [19]. It is known that GQM provides a systematic approach towards software measurement via organization of relevant goals, questions, and metrics. The second approach is the use of the TMM*i* framework. The TMM*i* reference model specifies the test process area and provides means for controlling the scope of the goals in the OSS-TPA quality model. The attention in this paper is specifically on the aspects related to the Test Monitoring and Control process area of the TMM*i* framework.

The lack of time is among the frequently-cited reasons for organizations to not adopt Capability Maturity Model Integration (CMMI) [20].

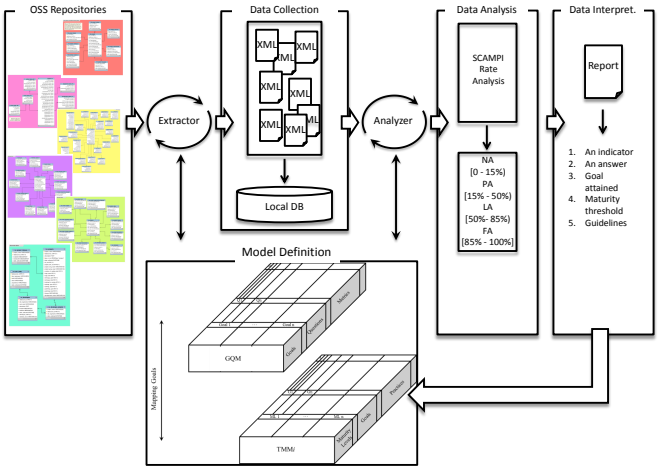

Fig. 1. A High-Level View of OSS-TPA Architecture

1) OSS-TPA Quality Model: Definition: The combination of GQM and TMM*i* contributes to decreasing the time for adopting a CMMI-based approach in an OSS project.
The measures for answering questions in the OSS-TPA quality model were computed manually, as well as, automatically, using a variety of tools.

The model definition module of OSS-TPA, as shown in Figure 2, consists of three abstract phases:

1. The conceptual phase, which derives the goal of managing the corrective actions to closure from the TMM*i* reference model.
2. The operational phase, which specifies a set of questions concerning the achievement of the goal stated in the conceptual phase. These questions are based on the relevant practices in the TMM*i* reference model.
3. The quantitative phase, which identifies a set of metrics for each specified question. These metrics are based on the work products associated with the test practices in the TMM*i* reference model.

It is known that appropriate corrective actions should be taken when test progress deviates significantly from the test plan, or product quality deviates from expectations [3].

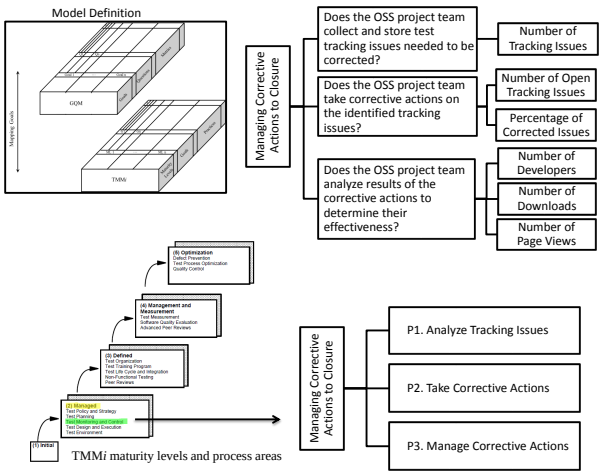

Fig. 2. A High-Level View of the OSS-TPA Quality Model

Metrics	Q_1	Q_2	Q_3
Number of Bugs	x		
Number of Patches	x		
Number of Feature Requests	x		
Number of Support Requests	x		
Number of Open Bugs		x	
Number of Closed Bugs		x	
Percentage of Corrected Bugs		x	x
Number of Open Feature Requests		x	
Number of Closed Feature Requests		x	
Percentage of Corrected Feature Requests		x	x
Number of Open Support Requests		x	
Number of Closed Support Requests		x	
Percentage of Corrected Support Requests		x	x
Number of Open Patches		x	
Number of Closed Patches		x	
Percentage of Corrected Patches		x	x
Number of Downloads			x
Number of Developers			x
Number of Page Views			x

TABLE II: Metrics of interest

Indeed, managing these actions to closure is one of the goals to be achieved in the Test Monitoring and Control process area that, in turn, belongs to Level 2 Managed in the TMM*i* framework, as shown in Figure 2.

2) OSS-TPA Quality Model: Measurement: The OSS-TPA quality model has a single goal:

Goal Manage corrective actions to closure with the aim of evaluating its maturity from a software manager's point of view.

To satisfy the aforementioned goal, the following questions are derived and formulated:

- Q_1 Does the OSS project team collect and store test tracking issues needed to be corrected?
- Q_2 Does the OSS project team take corrective actions on the identified tracking issues?
- Q_3 Does the OSS project team analyze results of the corrective actions to determine their effectiveness?

The OSS-TPA quality model includes three main quality attributes for managing corrective actions to closure, namely Analyze Issues, Take Corrective Action, and Manage Corrective Action, as shown in Figure 2. Each quality attribute is associated with a number of metrics. The selected metrics are based on the TMM*i* guidelines, and help in obtaining objective answers to the aforementioned questions.

The result, as shown in Table II, is a collection of 19 metrics, of which 4 correspond to the first quality attribute, 12 to the second quality attribute, and 3 to the third quality attribute. It can be noted that the mapping between the set of quality attributes and the set of metrics is not one-to-one.

The tracking issues in Table II are based on the classification scheme from SourceForge.net, and are categorized accordingly into four groups, namely Bugs, Feature Requests, Support Requests, and Patches.

B. Data Collection

The source of data for this study is the SourceForge Research Data Archive (SRDA) [21]. The SRDA is a repository of SourceForge OSS research data and allows the execution of SQL queries on tables exported from SourceForge.

The SQL query that follows has been used in this research to extract information on one of the tracking issues, namely bugs. For example, the following SQL query extracts the total number of bugs and the number of open ones for each project hosted on SourceForge in July 2010 for "sf0710" scheme:

```
SELECT g.group_id, ag.name, ac.count, ac.open_count
FROM sf0710.artifact_counts_agg ac, sf0710.artifact_group_list ag, sf0710.groups g
WHERE ac.group_artifact_id = ag.group_artifact_id AND g.group_id = ag.group_id
AND ag.name = 'Bugs'
```

The OSS projects with total issues (of the type bugs, feature requests, support requests, or patches) of zero have been excluded to get a valid number for the percentage of corrective issues (that is, total number of corrected issues divided by the total number of issues). Moreover, OSS projects that have assigned a NULL value to an issue have been excluded, as NULL cannot be considered as a valid number.

The data analysis module, as shown in Figure 1, deals with interpreting the collected data involving the percentage of issues for bugs, feature requests, and patches. This follows the ISO/IEC 15939 methodology for specifying indicators [22], and the ISO/IEC 15504 policy for rating indicators [23]. Each corrective action is measured on a four-point rating scale as follows:

NA:	Not Achieved	[0% - 15%)
PA:	Partially Achieved	[15% - 50%)
LA:	Largely Achieved	[50% - 85%)
FA:	Fully Achieved	[85% - 100%]

OSS Factor	Description	Indicator	Concept
Downloads	Total number of downloads of the software package	Moving from alpha to beta to stable; Achieved identified goals	Physical attribute; Community attribute
Developers	Total number of developers on the project	Activity level; User contribution; Knowledge sharing	Community attribute
Page Views	Total number of views of any of the project's website	User acceptance	Physical attribute; Community attribute

TABLE III: OSS Success/Abandonment Factors

C. Data Analysis and Data Interpretation

This section aims to answer the questions Q_1, Q_2, and Q_3 from Section III-A2. Figure 3 shows data related to the four tracking issues, and whether the data related to those issues was collected in an OSS project. For example, bugs were collected and stored in 30,029 out of 335,562 OSS projects.

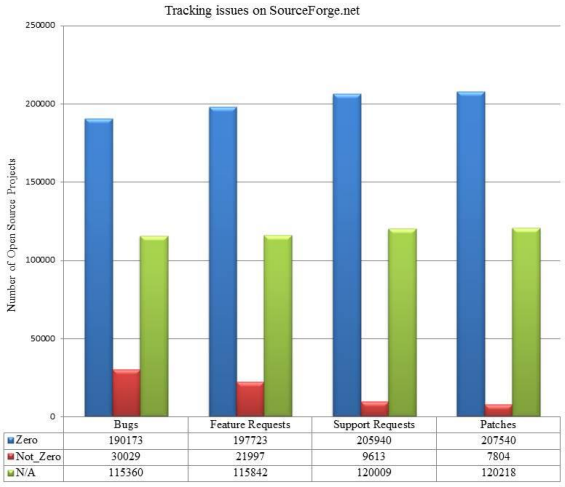

Fig. 3. Analyzing Tracking Issues

To answer the question Q_2 from Section III-A2, the following was carried out. Figure 4 shows the SCAMPI rating of OSS projects hosted on SourceForge.net. For example, 140 out of 1253 OSS projects failed to take corrective actions towards fixing bugs, while 411 out of 1253 OSS projects took corrective actions towards fixing bugs.

To answer the question Q_3 from Section III-A2, the following was carried out. The dataset consisting of 1253 OSS projects was categorized into four quartiles of 313 to 314 projects each. These were subsequently arranged in an ascending order of the applied factor of success, namely Number of Downloads, Number of Developers, and Page Views, as shown in Table III. These independent factors apply to the OSS development process [24], and allow distinguishing between successful and abandoned OSS projects (for the majority of those) in the SourceForge repository [25].

Next, assuming that the data collected is randomly distributed and is not normal, chi-square test [26] has been applied to determine the effectiveness of the corrective actions by investigating the dependency between these actions and the success factors (specified later in Table V).

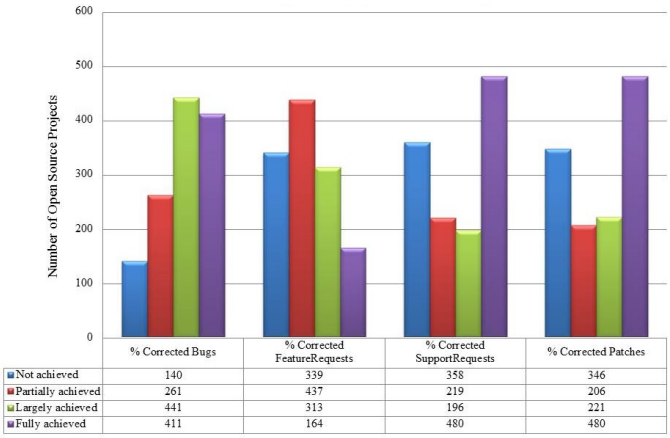

Fig. 4. Taking Corrective Actions

OSS Quart.	Frequencies	Not achieved	Partially achieved	Largely archived	Fully achieved	Totals
First	Observed	19	53	128	114	314
	Expected	35.08	65.41	110.51	103.00	
Second	Observed	42	72	103	96	313
	Expected	34.97	65.20	110.16	102.67	
Third	Observed	31	73	112	97	313
	Expected	34.97	65.20	110.16	102.67	
Fourth	Observed	48	63	98	104	313
	Expected	34.97	65.20	110.16	102.67	
Totals		140	261	441	411	1253

TABLE IV: Results of Chi-Square Test of Significance

The following conclusions can be drawn from the chi-square test results, as shown in Table IV, at a 5% level of significance.

There is insufficient evidence to conclude that the maturity of corrective actions towards fixing bugs, p-value > 0.05, is dependent on the success of OSS, using the number of downloads and the number of page views as success factors.

There is sufficient evidence to conclude that the maturity of corrective actions towards fixing bugs, p-value < 0.05, and the success of OSS, using the number of developers as a success factor, is interdependent.

There is sufficient evidence to conclude that the maturity of corrective actions towards fixing feature requests, support requests, and patches, p-value < 0.05, are dependent on the success of OSS, using the number of downloads, the number of developers, and the number of page views as success factors.

Tracking issues	Success factor	Chi-Square	P-Value	Dependency
Bugs	Number of downloads	13.938	0.125	No
Feature Requests	Number of downloads	20.170	0.017	Yes
Support Requests	Number of downloads	24.310	0.004	Yes
Patches	Number of downloads	31.607	0.000	Yes
Bugs	Number of developers	24.705	0.003	Yes
Feature Requests	Number of developers	25.969	0.002	Yes
Support Requests	Number of developers	35.291	0.000	Yes
Patches	Number of developers	49.881	0.000	Yes
Bugs	Number of page views	04.925	0.841	No
Feature Requests	Number of page views	20.469	0.015	Yes
Support Requests	Number of page views	29.262	0.001	Yes
Patches	Number of page views	43.866	0.000	Yes

TABLE V: Results of Chi-Square Test of Significance

IV. Support for OSS-TPA

The results of the previous section are supported by another empirical study [27]. In this study, six major OSS are studied to set up a reliability model using the general Weibull distribution. It is shown that widely-used measures, such as page views and downloads, are not highly correlated with the monthly bug arrival rate.

The results of the previous section also confirm the observations made in a study that has been applied on two major OSS projects, namely the Apache Web server and the Mozilla Firefox Web browser [28], namely that most bugs were reported by a relatively small developer community and not end-users. This signifies that, for most OSS projects, the number of bugs is not highly dependent on the number of page views or on the number of downloads.

V. Directions for Future Research

The work presented in this paper can be extended in a few different directions.

For example, the OSS-TPA framework can benefit from the support of further empirical studies. In particular, investigating the maturity of the OSS test processes in repositories other than SourceForge.net and/or with different process areas of the TMM*i* framework, is of research interest.

The list of success factors for OSS projects stated in this paper is not fixed, and can evolve. Indeed, other success factors might result in a different perspective on the OSS test process, and thereby constitutes another possible avenue of research interest.

VI. Conclusion

The growing number of competing software systems pose a challenge for their prospective adopters, and OSS are no different. The visibility of steps taken towards assuring the quality of an OSS is one of the most important factors towards its selection as a potential candidate.

This paper builds a foundation for evaluating and improving the OSS test process. In doing so, it presents a practical, customizable, and extensible framework, namely OSS-TPA, for understanding the management of corrective actions to closure in OSS.

The OSS-TPA framework supports the evaluation of a number of activities inherent in OSS test process, such as analyzing tracking issues, taking corrective actions, and managing corrective actions to closure, as shown by an empirical study presented in this paper. Using variations of GQM, the OSS-TPA framework can be customized to analyze the OSS test process in different contexts. Finally, OSS-TPA can be extended and applied to different maturity levels and relevant process areas of the TMM*i* framework.

VII. Acknowledgment

The authors would like to thank Olga Ormandjieva for discussions and comments on an earlier version of the paper.

References

[1] S. Morasca, D. Taibi, and D. Tosi, "Towards Certifying the Testing Process of Open-source Software: New challenges or Old Methodologies?" in *The 31st International Conference on Software Engineering (ICSE09) - The 2nd Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development (FLOSS09)*. Vancouver, Canada: IEEE, 2009, pp. 25–30.

[2] O. Taipale, K. Smolander, and H. Kalviainen, "Finding and Ranking Research Directions for Software Testing," in *Software Process Improvement*, ser. Lecture Notes in Computer Science, I. Richardson, P. Abrahamsson, and R. Messnarz, Eds. Springer-Verlag Berlin Heidelberg, 2005, vol. 3792, pp. 39–48.

[3] E. van Veenendaal and J. Jaap, "Testing Maturity - Where Are We Today: Results of the first TMMi benchmark," *Testing Experience*, vol. 3, no. 3, pp. 72–74, 2012.

[4] T. Abdou, P. Grogono, and P. Kamthan, "A Conceptual Framework for Open Source Software Test Process," in *The 36th Annual Computer Software and Applications Conference (COMPSAC12) - The 4th IEEE International Workshop on Software Test Automation (STA12)*. Izmir, Turkey: IEEE, 2012, pp. 458–463.

[5] R. S. Kenett and E. R. Baker, *Software Process Quality: Management and Control*. Marcel Dekker, 1999.

[6] R. Swinkels, "A Comparison of TMM and Other Test Process Improvement Models." Technical Report, Frits Philips Institute, Technische Universiteit Eindhoven, Netherlands, 2000.

[7] M. Rasking, "Experiences Developing TMMi as a Public Model," in *Communications in Computer and Information Science*, ser. Communications in Computer and Information Science, R. V. O'Connor, T. Rout, F. McCaffery, and A. Dorling, Eds. Springer-Verlag Berlin Heidelberg, 2011, vol. 155, pp. 190–193.

[8] International Organization For Standardization, "ISO/IEC WD 29119-2:2010 - Software and Systems Engineering - Software Testing - Test Process," 2010.

[9] N. Nagappan, L. Williams, M. Vouk, and J. Osborne, "Using In-Process Testing Metrics to Estimate Post-Release Field Quality," in *The 18th IEEE International Symposium on Software Reliability (ISSRE07)*. Trollhattan, Sweden: IEEE, 2007, pp. 209–214.

[10] I. Burnstein, *Practical Software Testing: A Process-oriented Approach*. Springer New York, 2003.

[11] D. Taibi, L. Lavazza, and S. Morasca, "OpenBQR: A Framework for the Assessment of OSS," in *Open Source Development, Adoption and Innovation*, ser. IFIP The International Federation for Information Processing, J. Feller, B. Fitzgerald, W. Scacchi, and A. Sillitti, Eds. Springer Boston, 2007, vol. 234, pp. 173–186.

[12] J.-C. Deprez and S. Alexandre, "Comparing Assessment Methodologies for Free/Open Source Software: OpenBRR and QSOS," in *Product-Focused Software Process Improvement*, ser. Lecture Notes in Computer Science, A. Jedlitschka and O. Salo, Eds. Springer-Verlag Berlin Heidelberg, 2008, vol. 5089, pp. 189–203.

[13] I. Samoladas, G. Gousios, D. Spinellis, and I. Stamelos, "The SQO-OSS Quality Model: Measurement Based Open Source Software Evaluation," in *Open Source Development, Communities and Quality*, ser. IFIP International Federation for Information Processing, B. Russo, E. Damiani, S. Hissam, B. Lundell, and G. Succi, Eds. Springer Boston, 2008, vol. 275, pp. 237–248.

[14] C. Ardagna, E. Damiani, and F. Frati, "FOCSE: An OWA-based Evaluation Framework for OS Adoption in Critical Environments," in *Open Source Development, Adoption and Innovation*, ser. IFIP International Federation for Information Processing, J. Feller, B. Fitzgerald, W. Scacchi, and A. Sillitti, Eds. Springer Boston, 2007, vol. 234, pp. 3–16.

[15] M. Michlmayr, "Software Process Maturity and the Success of Free Software Projects," in *Software Engineering: Evolution and Emerging Technologies*, K. Zieliski and T. Szmuc, Eds. IOS Press Amsterdam, The Netherlands, 2005, pp. 3–14.

[16] G. Gousios, V. Karakoidas, K. Stroggylos, P. Louridas, V. Vlachos, and D. Spinellis, "Software Quality Assessment of Open Source Software," in *The 11th Panhellenic Conference on Informatics (PCI07)*, Patras, Greece, 2007, pp. 303–315.

[17] M. Cabano, C. Monti, and G. Piancastelli, "Context-Dependent Evaluation Methodology for Open Source Software," in *Open Source Development, Adoption and Innovation*, ser. IFIP The International Federation for Information Processing, J. Feller, B. Fitzgerald, W. Scacchi, and A. Sillitti, Eds. Springer New York, 2007, vol. 234, pp. 301–306.

[18] Y. Wang and G. A. King, *Software Engineering Processes: Principles and Applications*. CRC Press, 2000.

[19] V. Basili, G. Caldiera, and D. H. Rombach, "The Goal Question Metric Approach," in *Encyclopedia of Software Engineering*, J. Marciniak, Ed. Wiley, 1994.

[20] M. Staples, M. Niazi, R. Jeffery, A. Abrahams, P. Byatt, and R. Murphy, "An Exploratory Study of Why Organizations Do Not Adopt CMMI," *Journal of Systems and Software*, vol. 80, no. 6, pp. 883–895, 2007.

[21] G. Madey, "The SourceForge Research Data Archive, SRDA, University of Notre Dame," 2010.

[22] International Organization For Standardization, "ISO/IEC 15939:2007-Systems and Software Engineering - Measurement Process," 2007.

[23] H. V. Loon, *Process Assessment and ISO/IEC 15504: A Reference Book*. Springer New York, 2007.

[24] K. Crowston, H. Annabi, J. Howison, and C. Masango, "Towards a Portfolio of FLOSS Project Success Measures," in *The 26th International Conference on Software Engineering (ICSE04) - Collaboration, Conflict and Control: The 4th Workshop on Open Source Software Engineering*, 2004, pp. 29–33.

[25] C. M. Schweik, R. C. English, and S. Haire, "Factors Leading to Success or Abandonment of Open Source Commons: An Empirical Analysis of Sourceforge.net Projects," *South African Computer Journal*, vol. 43, pp. 58–65, 2009.

[26] R. R. Johnson and P. J. Kuby, *Elementary Statistics*. Brooks/Cole, 2007.

[27] Y. Zhou and J. Davis, "Open Source Software Reliability Model: An Empirical Approach," *The 27th International Conference on Software Engineering (ICSE05) - Open Source Application Spaces: The 5th Workshop on Open Source Software Engineering*, vol. 30, no. 4, pp. 1–6, 2005.

[28] A. Mockus, R. T. Fielding, and J. D. Herbsleb, "Two Case Studies of Open Source Software Development: Apache and Mozilla," *The ACM Transactions on Software Engineering and Methodology*, vol. 11, no. 3, pp. 309–346, 2002.