Electronic Communications of the EASST
Volume 48 (2012)

Proceedings of the
Fifth International Workshop on on Foundations
and Techniques for Open Source Software Certification
(OpenCert 2011)

Analysis of Collaboration Effectiveness and Individuals' Contribution in FLOSS Communities

Antonio Cerone, Simon Fong and Siraj Ahmed Shaikh

15 pages

Guest Editors: Luís Soares Barbosa, Dimitrios Settas
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Analysis of Collaboration Effectiveness and Individuals' Contribution in FLOSS Communities

Antonio Cerone[1], **Simon Fong**[2] **and Siraj Ahmed Shaikh**[3]

[1] antonio@iist.unu.edu,
UNU-IIST — International Institute for Software Technology
United Nations University, Macau SAR China

[2] ccfong@umac.mo
Department of Computer and Information Science
University of Macau, Macau SAR China

[3] s.shaikh@coventry.ac.uk
Department of Computing, Coventry University, UK

Abstract: Free/Libre Open Source Software (FLOSS) development has proven itself over the years to be able to deliver high-quality software products. However, it is not clear how quality emerges from the large amount of loosely organised activities of a FLOSS community. This makes it difficult to apply traditional quality metrics and certification processes to FLOSS products.

This paper investigates possible indicators of collaboration effectiveness and quality of individuals' contribution that could be extracted from the data available in repositories of FLOSS projects. The ultimate purpose of this effort is to develop quantitative metrics for these indicators and merge such metrics into a global metric for FLOSS software quality to be used in a certification process.

Keywords: Open Source Software, software quality, collaboration models, trust models, certification.

1 Introduction

Free/Libre Open Source Software (FLOSS) development has proven itself over the years to be able to deliver high-quality software products. Moreover, the potential benefits of the FLOSS development model include "the ability to more easily carry out open peer reviews, add new functionality either locally or to the mainline products, identify flaws, and fix them rapidly — for example, through collaborative efforts involving people irrespective of their geographical locations and corporate allegiances." [Neu05].

Although the high-quality of a number of Free/Libre Open Source Software (FLOSS) products has been accepted as a fact, it is still unclear how such high-quality emerges from the "bazaar-style" activities of a FLOSS community. Raymond claims that "The high level of quality of free software is partly due to the high degree of peer review and user involvement" [Ray99]. McConnell [McC99] acknowledges the efficiency of extensive field testing and peer review, along with an emphasis on the need for a comprehensive methodology for open source development.

This is important if open source development is to be used for producing high quality complex software for use in critical domains such as safety and security. In fact, Schneider [Sch00] finds that open source software still falls short of requirements for security systems. Halloran and Scherlis [HS02] review a number of notable quality practices on some popular open source projects, of which good project communication and management is highlighted.

Coverity has been analysing the quality of open source software since 2006, using Coverity Scan, a tool for automated static analysis of source code [Cov]. The results of this analysis have been published by Coverity in annual reports. The 2008 and 2009 reports show that defect density has fallen by 16% over the period 2006–2008 to the extent that the static analysis defect density averaged across all the participating projects is 0.25, or roughly one defect per 4,000 lines of code [Cov08, Cov09]. Improvements to Coverity Scan and its underlying technology over the past years have allowed to flag more defects than in prior years. As a consequence a direct comparisons to prior years Coverity Scan results is no longer possible. The 2010 report highlights that (1) nearly half (45%) of the defects discovered in open source are considered high-risk defects; (2) there has been very little change in the types of defects found and frequency in which they occur in open source software; and (3) open source accountability is fragmented [Cov10]. In spite of these drawbacks, the report expresses the expectation that as open source continues to mature, more and more projects will begin to adopt stronger quality practices. The 2011 report includes a comparison of proprietary software and open source software that leads to an important key finding: "Open source quality for active projects in Coverity Scan is better than the software industry average" [Cov11].

Large communities of users have been growing around popular high-quality FLOSS products such as Linux, Ubuntu, Apache, MySQL and Moodle, among the others. The widespread use of FLOSS products not only involves personal users, who install Linux/Ubuntu and MySQL on their machines, but also small and medium enterprises, who use Apache servers and FLOSS tools in their production activities or even incorporate FLOSS components in their software products, and academic and teaching institutions, who use FLOSS products in their research and educational activities, including Learning Management Systems (LMS) such as Moodle. More recently, large software companies have been launching FLOSS projects with the aim to get revenues by adopting a freemium business model, in which the basic product or service is provided free of charges, while a premium is charged for the provision of support services and/or advanced features and functionality.

As highlighted by the 2010 Coverity report [Cov10], due to the rapid adoption of open source as part of many commercial software supply chains, there is an increasing demand from OEMs (original equipment manufacturer) to get visibility into the open source software development process and hold open source to the same scrutiny as their other software systems to meet the enterprises necessary quality, safety, and security requirements. The fragmented nature of open source supply chain, made up of multiple components from multiple development teams, makes if difficult to identify who is accountable to upholding requirements and providing visibility and who is to be blamed if and when there is a problem [Cov10].

All the above considerations lead to a major limitation for the diffusion of FLOSS products: the lack of a certification process that could provide accountability, meet certification standards and facilitate the approval of a certification authority. However, the lack of accurate information on how quality emerges from the large amount of loosely organised activities of a FLOSS community makes it difficult to apply traditional quality metrics and certification processes to FLOSS products. For instance, if we consider McCall's production revision quality factors [MRW77], can we claim that a FLOSS product lacks *maintenability* because there are no defined coding standards and guidelines to which programming has adhered? The philosophy of freedom and absence of hierarchical organisation typical of FLOSS communities results in collaborative production environments in which there is no space for *prescriptive* standards and strict guidelines. Communication and collaboration are the drivers of such production environments and naturally determine the evolution of programming practices within teams of contributors and across the FLOSS community, even beyond a specific FLOSS project. In such a context, a *descriptive* approach that analyses the FLOSS community of practise and its activities is likely to define better indicators of the quality of the software product than a *prescriptive* approach that tries to check whether these activities follow prescribed standards and guidelines [Cer12].

In this paper, we carry out a preliminary discussion towards a methodology to analyse community activities in FLOSS projects and extract from the data collected pieces of semantic information to be used as indicators of the quality of the software product. In particular, we focus on two aspects of the FLOSS community of practice: collaboration effectiveness and quality of individuals' contribution.

In Section 2 we identify possible indicators of collaboration effectiveness and present *cognitive-based collaboration models* and a *filtering trust network model*, and discuss how to adapt them to a FLOSS context. Based on the adaptation of one of such cognitive-based collaboration models to a FLOSS context, Section 2.1 illustrates an example of metric to characterise collaboration effectiveness in a FLOSS community. Section 2.2 illustrates an instantiation of filtering trust network model and further discusses how to adapt such a model to a FLOSS context. Section 3 analyses engagement, productivity and reputation, as indicators to be used to define metrics that characterise quality of individuals' contribution. Finally Section 4 summerises the achievements of this work and discusses ideas and objectives for future work.

2 Collaboration Effectiveness

Collaboration within FLOSS communities is enabled by the usage of tools, such as versioning systems, mailing lists, reporting systems, etc. These tools also serve as repositories which can be data mined to understand the identities of the individuals involved in a communication, the topics of their communication, the amount of information exchanged in each direction, as well as the amount of contribution in terms of code commits, bug fixing, reports and documentation produced and email postings. Such a large amount of data can be selectively collected and then analysed not only by using inferential statistics to identify activity patterns but also by using ontology engineering formalisms that support the extraction of semantic information. In the area of Empirical Software Engineering, cyber-archeology [SSS07] has been applied to these repositories to learn and better understand the patterns of contribution of FLOSS developers in the projects concerned [SC10].

In such previous work [SC10] data collection has involved communications mainly in terms of participants, quantity and sometimes topics but neglected the objective collection of actual communication contents. At most, content data has been collected through questionnaires and

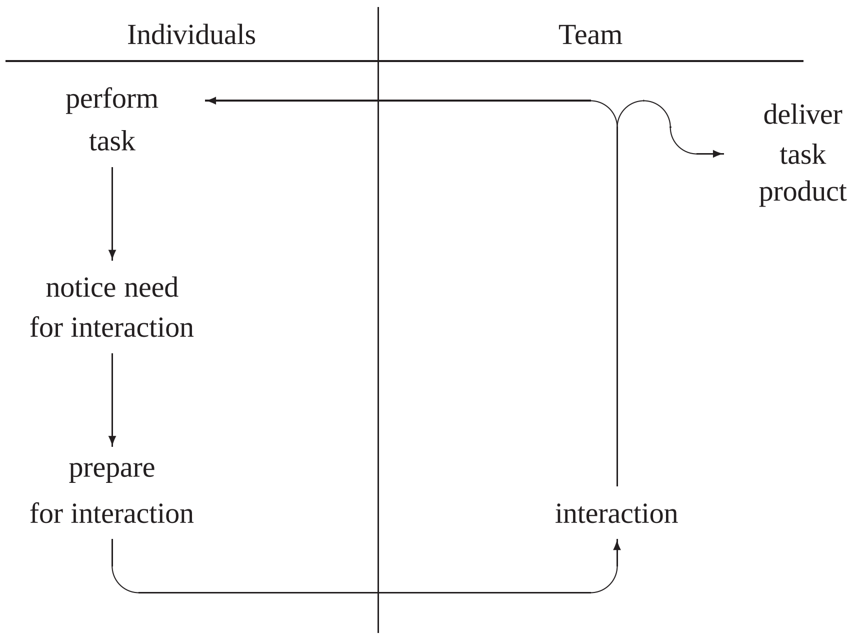

Figure 1: Noble and Letsky Individual Team interplay.

surveys or through written reports by researcher who joined the community as observers, thus providing subjective rather than objective data. With reference to existing cognitive-based collaboration models [NL02], data mining methods can be used to extract content information from email communication and posting with the support of appropriate ontologies aiming to identify patterns, progress, evolution and achievements in the collaboration process occurring within groups of participants. This requires the analysis of incremental data and the construction and analysis of graph data from the overall social networking aspect of the FLOSS community.

Cognitive-based collaboration theory [NL02] aims to consider many different factors underlying the mechanisms that connect community member understandings to community effectiveness in production. There is no single model that represents all of these factors, but separate models that address different factors. Only some of these models are relevant to a self-organising non-hierarchical community as is a FLOSS community.

The individual-team interplay model [NL02] described in Figure 1 is a cyclic model in which individuals *perform a task*, *notice need for interaction*, *prepare for interaction*, *perform the interaction* and *go back to the task*, possibly *delivering the product of the task* and then starting a new task. Figure 2 shows how this model can be adapted to the FLOSS context. Here interaction consists essentially in posting activities while the product of the task is delivered through a commit activity by the individual or through an approval or release decision by the leader team. The interaction process is decomposed into two looping sub-processes [Cer12]:

learning sub-process in which the exchange of knowledge between individual and community results in the growth of knowledge at both the individual level and the team or community

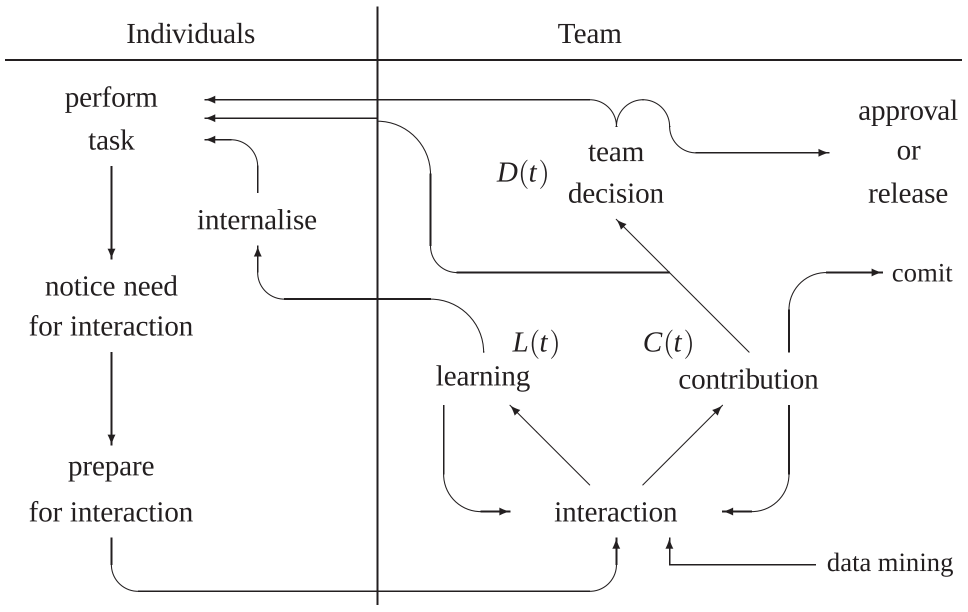

Figure 2: Individual Team interplay for FLOSS.

level;

contribution process in which a contribution in terms of commit of code, bug report, etc. is the result of an exchange of communications.

With reference to this adapted model, Section 2.1 illustrates a possible way for defining a metric to characterise collaboration effectiveness in a FLOSS community.

The Cognition-Behaviour-Product model [NL02] emphasises the nature of the relationship between individual and team understandings, individual and team behaviours and individual and team production. In a FLOSS context, this model has the important role to explain how task quality and understanding affect each other and is essential in measuring individual task performance and collaboration effectiveness.

An important factor in collaboration is trust. Fong et al. [CKF12, CF10], have developed a *filtering trust network model* to study about fusing elusive information and deriving trust factors in a social network, by taking Facebook as a case.

The above models can be used to define metrics for information interaction, task performance, product quality, peer trust [NL02, CF10]. In addition to metrics definition, some contextual inferring mechanisms are needed as a data pre-processing step for extracting the semantics and essences from the empirical data using machine learning techniques. Techniques previously applied to the analysis of public moods based on Internet comments [Fon12] can be also applied to implement such mechanisms. This will be further discussed in Section 2.2.

2.1 Towards an Individual-Team Interplay Metric

One way for defining a metric to characterise collaboration effectiveness within a community that is continuously evolving and producing, as a FLOSS community, is to take snapshots of the collaboration measure at consecutive intervals of a sufficiently short duration Δt and then calculate the average of the measure over the entire community lifetime.

With reference to Figure 2 we consider the numbers

- $L(t)$ of learning activities,
- $C(t)$ of contributions,
- $D(t)$ of team decisions

that occur during the period $[t, t + \Delta t]$ for a given time t. It has been observed [Cer12] that participation in a FLOSS project evolves through three stages:

Stage 1 (understanding) in which communication is heavily used to capture, describe and understand contents, while no production activity is performed;

Stage 2 (practice) in which the role of communication gradually moves to the proposal of new contents, the defence of the proposed contents and the criticism to existing contents or contents proposed by others, while production activity starts as a trial and error process;

Stage 3 (developing) in which real development occurs.

Let

- $N_1(t)$ be the number of contributors in the understanding stage (stage 1),
- $N_2(t)$ be the number of contributors in the practice stage (stage 2),
- $N_3(t)$ be the number of contributors in the developing stage (stage 3)

at time t. Then the measure of the collaborative effectiveness that characterises the *quality enhance* of the project development during the period $[t, t + \Delta t]$ for a given time t as follows.

$$M_t = \frac{L(t) \cdot K_L}{N_1(t) + N_2(t)} + \frac{C(t) \cdot K_C}{N_2(t) + N_3(t)} + \frac{D(t) \cdot K_D}{N_3(t)} \quad \text{if} \quad N_2(t) \neq 0 \quad \text{and} \quad N_3(t) \neq 0.$$

where K_L, K_C and K_D are constants. This measure shows how collaboration effectiveness in individual-team interplay within a FLOSS community is characterised by a combination of learning activities ($L(t)$) by individuals at the learning ($N_1(t)$) and practice ($N_2(t)$) stages, contribution activities ($C(t)$) by individuals at the practice ($N_2(t)$) and developing ($N_3(t)$) stages, and team decision activities only by individuals at developing ($N_3(t)$) stages. The contribution of each of the three categories of activities to the quality enhance is directly proportional to the number of activities of that category during the considered time period and inversely proportional to the number of individuals that characterise that category of activities. The constants give weights to the three categories of activities. It is reasonable to expect that $K_L < K_C < K_D$. Note that $N_3(t) > 0$ for

$N(t)$	$N_1(t)$	$N_2(t)$	$N_3(t)$	$L(t)$	$C(t)$	$D(t)$	M_t
10	6	2	2	5	1	1	3.125
10	5	1	4	5	1	1	2.233
10	3	3	4	5	1	1	2.219
10	5	1	4	4	2	1	2.467
10	3	3	4	3	1	3	3.786

$$K_L = 1 \qquad K_C = 2 \qquad K_D = 4$$

Figure 3: Example.

any active project and if $N_1(t) = N_2(t) = 0$ the equation above may be modified by removing the first addend.

Let us consider the example in Figure 3 where we use constants $K_L = 1 < K_C = 2 < K_D = 4$. A project with 10 contributors, 6 at the learning stage, 2 at the practice stage and 2 at the developing stage with 5 learning activities, 1 contribution activity and 1 team decision activity has a quality enhance 3.125. As contributors move to more mature stages, if there is no variation on the activity pattern, the quality enhance decreases to 2.233 (2 contributors move to the developing stage) and then to 2.210 (to further contributors move to the practice stage). On the other hand, an increase in the number of contribution activities is associated with an increase in quality enhance from 2.233 to 2.467 (1 additional contribution) and an increase in the number of team decision activities is associated with an increase in quality enhance from 2.219 to 3.786 (2 additional team decisions).

The quality by development of a project of duration D is therefore defined as follows

$$M = \frac{1}{n} \sum_{t=0}^{n} M_t$$

where $n = D/\Delta t$.

2.2 Filtering Trust Network Model

Reputation is considered as one of the key parameters for defining the quality of an individual's contribution to a FLOSS project. For example a reputable contributor is expected to be an active and valuable contributor; making quality contributions over a period of time, and this will earn him a good reputation, and vice-versa. Without being able to easily gauge quality as it involves complex context and semantics (and is quite subjective), reputation is often related to trust which is a social evaluation or opinion about an individual person.

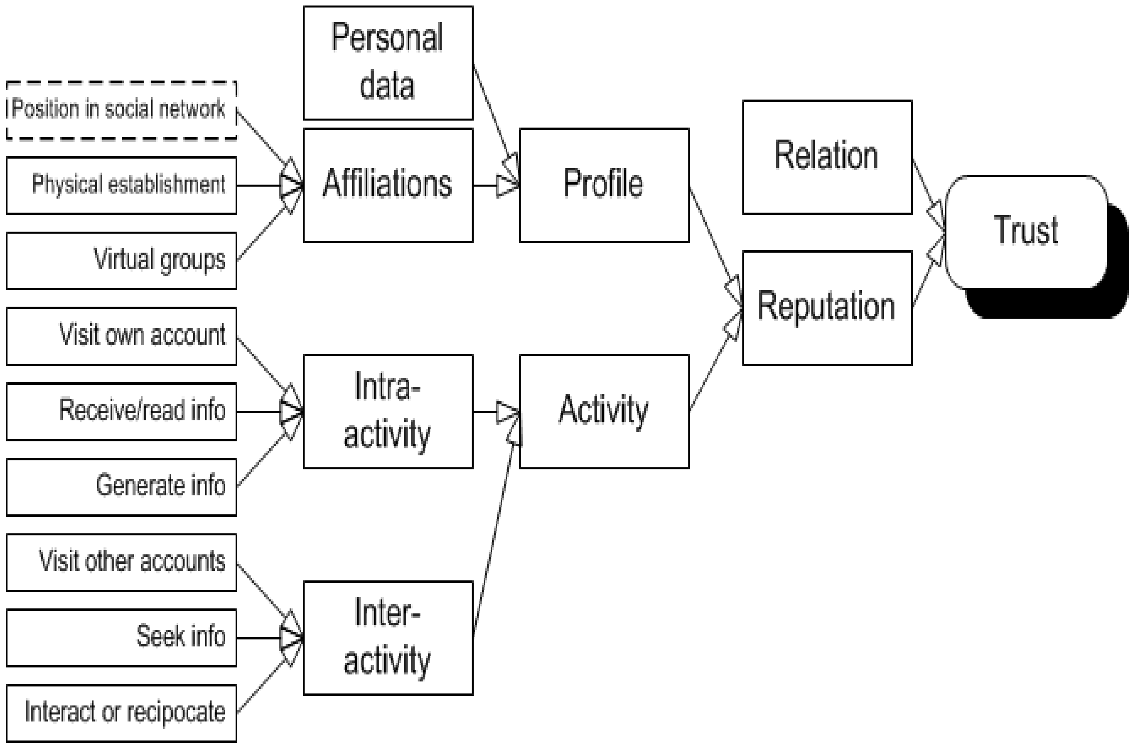

Figure 4: Hierarchical trust metrics model.

In a recent paper [CF10], online trust is inferred via a hierarchical metrics model which consists of explicit trust by relation, and implicit trust by reputation. Figure 4 shows the online trust metrics model. Trust by relation would have to be manually configured by a system administrator. Trust by relation suggests that a relatively high degree of trust is perceived between parents and children, spouses and siblings, and perhaps only a moderate amount of trust is reckoned for a stranger who was met on a trip. In a FLOSS project, it is more relevant to deduce implicit trust — hence reputation, from a series of online social criteria, because of the availability of the collected data.

Many techniques for inferring trust quantitatively exist in the literature ranging from psychology to computer science. It is noted, however, that the selection of factors or criteria to account for and quantify in the trust equation is not definitive. In fact, there is no definite selection of trust factors ever published or consistently agreed by all researchers. A growingly popular set of factors for deducing online trust is the one proposed by Massa and Bhattacharjee [MB04], which takes into account the number of messages exchanged between users, the number of mutual readings and comments on each others blogs (or walls in a social network), and the number of common chats within a specified period. Some of these attributes are included in a typical trust filtering equation as shown in the following model instantiation as an example. The equation was used to empower a collaborative filtering algorithm for implementing a social network-based recommender [CKF12].

The quantified trust by reputation $T_{a,u}$ between user a and user u can be evaluated based on the measured values of these attributes using Multiple Attribute Utility Theory (MAUT) as follows:

$$T_{a,u} = \alpha\frac{\sum PI}{tot_{PI}} + \beta\frac{\sum WTWP}{tot_{WTWP}} + \gamma\frac{\sum LF}{tot_{LF}} + \delta\frac{\sum NF}{tot_{NF}} + \varepsilon\frac{\sum T}{tot_{T}} + \theta\frac{\sum GIC}{tot_{GIC}}$$

where $\alpha + \beta + \gamma + \delta + \varepsilon + \theta = 1$, and

- PI quantifies personal information;
- $WTWP$ is the number of wall-to-wall posts;
- LF is the number of links among friends;
- NF is the number of friends;
- T is the number of tags between user a and user u;
- GIC is the number of groups in common.

and in each term, the denominator tot represents the total quantity of that particular attribute of this user.

Given the trust estimation equation between any pair of users, a quantifying reputation R_a could be defined as the general perceivable credibility of a particular user a by all his peers

$$R_a = \phi \sum_{\substack{u=0 \\ u\neq a}}^{N} T_{a,u}$$

where N is the total number of users in the online community and ϕ is the normalizing factor. However, it should be clear that the selection of criteria is tightly dependent on the functionalities that each specific social networking site provides to its users, and thus can vary greatly from site to site. A recent study [GK09] generalized the trust factors into seven categories (called time strength dimensions), and has statistically shown how useful they are as predictive variables for trust in social networks. These factors are illustrated in Figure 5.

In addition to the metrics mentioned above, which can be extracted directly from the users' accounts, the influence of a particular user in a social network can be quantified by how far and how deep his messages propagate to. It is also known that the reputation of a good user is partially measured by his popularity and that of his messages, assuming they are of good-will. Klout [Klo], the standard for measuring online influence, has derived a collection of in-depth metrics as social media analytics that measure the outreach level and popularity of one's messages; hence they may form insightful indicators for inferring one's reputation in his social group. Klout scores which range from 1 to 100 correspond to the assessment of the three following measures:

True Reach which measures the group size of the user's engaged followers who actively respond to his posted messages;

Amplification Propensity which calculates the likelihood that the user's posted messages will invite reciprocating messages;

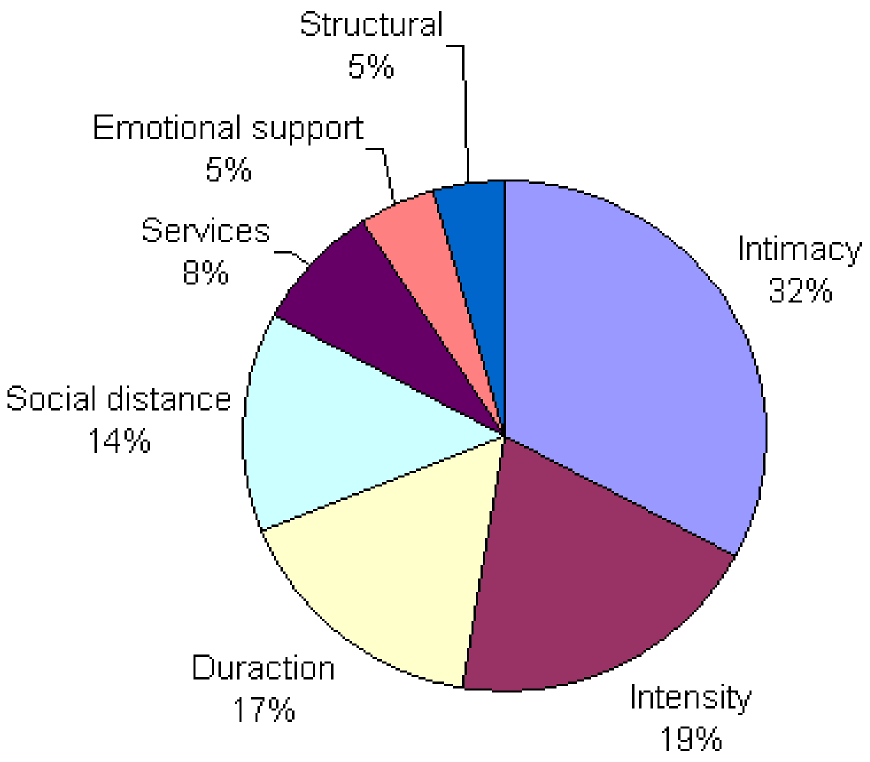

Figure 5: The distribution of the predictive power of the seven time strength dimensions as part of the how-strong model. Source: Gilbert and Karahalios [GK09].

Network Score which computes the influential impact value on the users followers.

So far it is observed that all the social media metrics proposed by the other researchers are either focused on the quantity of one's generated messages and his received messages (the so-called interactions), or his intimacy (relations) with his peers. Such metrics are indeed pertinent for social network analysis because a social network essentially aims to provide a convergent platform for social activities. Such metrics, however, may not suffice the inference of reputation, at least not directly or easily, in FLOSS environment because FLOSS activities are usually task-oriented with specific objectives. For example, an influencer in a social network may be a propaganda promoter which does little constructive contribution to a specific project but counter-productive noise, perhaps. The current models lack of some measures that indicate or imply who has contributed how much a share of progress in a collaborative environment like FLOSS. Though the social media metrics and the trust filtering model may serve as good ingredients for data mining and measuring the interaction level respectively, a higher level of processing may be needed for adding the semantics and contextual references to the messages that were exchanged.

3 Quality of Contributions

Quality of an individual's contribution to a FLOSS project can be measured in terms of three parameters: engagement, productivity and reputation.

Shaikh and Cerone [SC09] have identified some factors that are unique to the FLOSS development process and influence the entire software development process and, consequently, the quality of the final software product. In their work, Shaikh and Cerone also define an initial framework in which such factors can be related to each other and to the quality. In particular, they distinguish three main notions of quality in the context of FLOSS development

quality by access which aims to measure the degrees of availability, accessibility and readability of source code in relation to the media and tools used to directly access source code and all supporting materials such as the documentation, review reports, testing outcomes, as well as the format and structural organisation of both source code and supporting materials.

quality by development which aims to measure the efficiency of all development and communication processes involved in the production, evolution and release of source code, its execution, testing and review, as well as bug reporting and fixing;

quality by design which corresponds to the traditional notion of software quality [Pre00]: the end quality is judged by the design and implementation of the actual software and the code that underlies it.

These three notions of quality can be used as a basis for characterising the quality of engagement of an individual in the community. Every activity of an individual can be classified under an appropriate category of quality and marked to contribute to the final software product accordingly. Bug reporting, testing and reviews enhance quality by development, the media and format used to externalise such contributions affect quality by access, whereas evidence of planning and design, and validation of software code contribute to quality by design.

Number of commits and communications provide indicators of the level of contribution of an individual in the community and have been statistically analysed to determine patterns of contribution and their implications for the quality of code [SC10]. The number of commits describes how much the individual delivers in terms of product and is therefore an indicator of the individual's productivity. Although there is no guarantee on the quality of the product delivered, number of commits can be considered by itself an important parameter in evaluating the quality of the individual as a contributor. Moreover, by integrating data on the quality of the contribution in terms of quality of code and bug reports produced, and efficacy of bug fixing, and quantitative data on the approval and inclusion of the resultant artifacts in a release by the project leader team, we can define a more accurate measure of the quality of the individual's productivity.

Evidence suggests [RHS06] that reputation serves to be a major source of motivation for developers to participate in a community. There is also a clear link between higher status of developers' reputation and higher levels of income [HNH03], which makes it even more significant given the desire for career progression for developers as another motivation to participate in

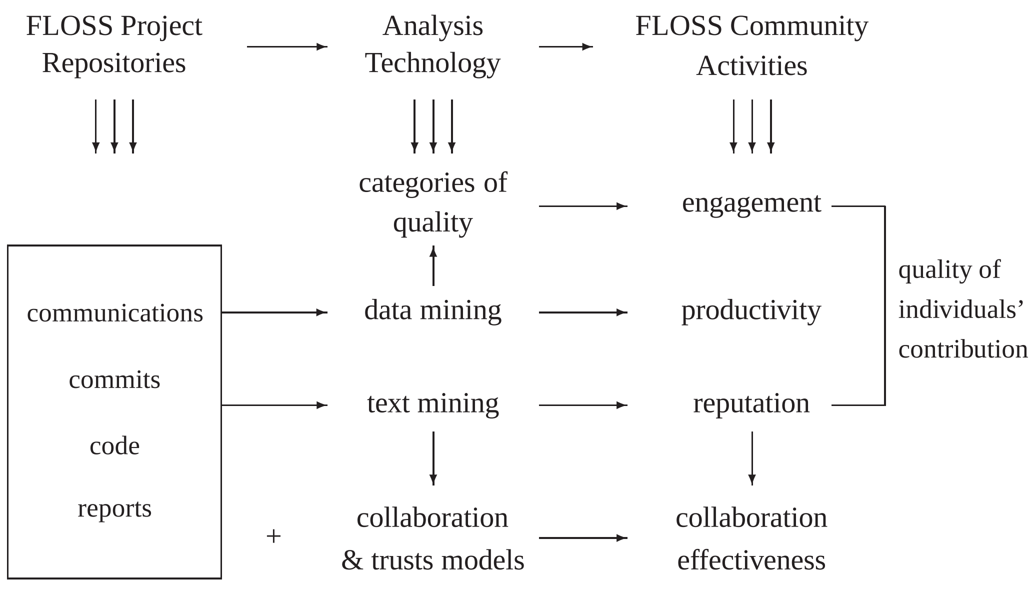
FLOSS Project Repositories
Analysis Technology
FLOSS Community Activities
categories of quality
engagement
quality of individuals' contribution
communications
commits
code
reports
data mining
productivity
text mining
reputation
+
collaboration & trusts models
collaboration effectiveness

Figure 6: Descriptive approach for FLOSS quality assurance

FLOSS communities. Communications among members of a FLOSS community can be analysed to extract information about the reputation an individual has achieved within the community. Text mining of communications can be used to identify keywords and phrases that may indicate whether an individual is asking or providing support and whether an answer or suggestion is taken on board or refuted by others. In addition, the filtering trust network model [CKF12, CF10] discussed in Section 2.2 can highlight trust factors that contribute to build an individuals reputation. Factors influencing individual reputation can then be categorised as both computational (such as communications) and non-computational (trust factors, ratings).

The reputation of an individual depends not just on the participation to a specific FLOSS project, but on the global activities of that individual in the FLOSS world. Therefore reputation information of a given individual have to be collected over all FLOSS projects listing that individual as a participants and be integrated with personal information including the individual's background, publication in the FLOSS field and participation in related social networks and discussion fora. Finally, the level of engagement of an individual within a project is visible to the entire project community and, therefore, implicitly affects that individual's reputation.

4 Conclusion and Future Work

We have discussed possible indicators of collaboration effectiveness and quality of individuals' contribution which can be extracted from the data available in repositories of FLOSS projects. The approach is summarised in Figure 6. Data involving communications, commits, code and reports may be collected from repositories of FLOSS projects and analysed using data mining and text mining technology, collaboration and trust models and applying the categories of quality defined by Shaikh and Cerone [SC09], to extract indicators of specific FLOSS community activities. Such indicators can be used to define metrics that characterise collaboration effectiveness in terms of information interaction, task performance, product quality and peer trust, as we have discussed in Section 2, and quality of individuals' contribution in terms of engagement, productivity and reputation, as we have discussed in Section 3.

We have seen in Section 2.1 how to use the individual-team interplay model for FLOSS to define a metric to characterise collaboration effectiveness in a FLOSS community. We have illustrated in Section 2.2 an instantiation of filtering trust network model and further discussed how to adapt such a model to a FLOSS context as a characterisation of reputation and as a further characterisation of collaboration effectiveness.

Finally, we have discussed in Section 3 how separate metrics can be defined to characterise engagement, productivity and reputation of individuals. However, it is still unclear how to combine such metrics into a global metric that could quantify the quality of an individuals contribution to a specific FLOSS project. One of the challenges is represented by possible interrelations between the three metrics; for instance, we have pointed out above that engagement affects reputation. In addition to trust, recent work in this area points to various other social parameters that affect reputation [HZC12] demonstrating how reputation obeys the laws of cumulative advantage (through higher likelihood of attracting good reputation given past reputation) and homophily (providing advantage through shared affiliations in terms of common FLOSS projects). Further work will aim to

1. analyse all these various social parameters that affect reputation and their impact on quality of individual's contribution and collaboration effectiveness;
2. combine separate metrics for quality of individuals' contribution possibly into a contributor profile to offer a more member-centric view of FLOSS development than the community-centric analysis traditionally common in this domain.

In our future work, as the final objective of the work presented in this paper, we intend to merge all metrics defined for specific quality indicators into a comprehensive framework to determine a global metric for FLOSS software quality to be used in a certification process.

Acknowledgements: This work has been supported by Macao Science and Technology Development Fund, File No. 019/2011/A1, in the context of the PPAeL project.

Bibliography

[Cer12] A. Cerone. Learning and Activity Patterns in OSS Communities and their Impact on Software Quality. In *Proceedings of OpenCert 2011*. Volume 48 of Electronic Communications of the EASST. 2012.

[CF10] W. Chen, S. Fong. Social Network Collaborative Filtering Framework and Online Trust Factors: a Case Study on Facebook. In *The 5th International Conference on Digital Information Management (ICDIM 2010)*. July 2010, Thunder Bay, Canada, pp. 266–273. IEEE Press, 2010.

[CKF12] W. Chen, R. Khoury, S. Fong. Web 2.0 Recommendation Service by Multi-Collaborative Filtering Trust Network Algorithm. *Journal of Information Systems Frontiers*, 2012.

[Cov] Coverity Scan.
http://scan.coverity.com/about.html

[Cov08] 2008 Coverity Scan Open Source Report. 2008.
http://scan.coverity.com/report/

[Cov09] 2009 Coverity Scan Open Source Report. 2009.
http://scan.coverity.com/report/

[Cov10] Coverity Scan: 2010 Open Source Integrity Report. 2010.
http://www.coverity.com/library/pdf/coverity-scan-2010-open-source-integrity-report.pdf

[Cov11] Coverity Scan: 2011 Open Source Integrity Report. 2011.
http://www.coverity.com/resource-library

[Fon12] S. Fong. Measuring Emotions from Online News and Evaluating Public Models from Netizens Comments: A Text Mining Approach. *Journal of Emerging Technologies in Web Intelligence* 4(1):60–66, February 2012.

[GK09] E. Gilbert, K. Karahalios. Predicting Tie Strength With Social Media. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems (CHI 09)*. April 4–9, 2009, Boston, Massachussetts, USA, pp. 211–220. ACM, 2009.

[HNH03] G. Hertel, S. Niedner, S. Herrmann. Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel. *Research Policy* 32(7):1159–1177, 2003.

[HS02] T. J. Halloran, W. L. Scherlis. High Quality and Open Source Software Practices. In *2nd Workshop on Open Source Software Engineering*. May 2002.

[HZC12] D. Hu, J. L. Zhao, J. Cheng. Reputation management in an open source developer social network. *Decision Support Systems* 53(3):526–1058, Jun 2012.

[Klo] Klout.
http://www.klout.com

[MB04] P. Massa, B. Bhattacharjee. Using trust in recommender systems: An experimental analysis. In *Proceedings of the 2nd International Conference in Trust Management (iTrust 2004)*. Lecture Notes in Computer Science 2995, pp. 221–235. 2004.

[McC99] S. McConnell. Open Source Methodology: Ready for Prime Time? *IEEE Software* 16(4):6–8, Jul/Aug 1999. IEEE Computer Society.

[MRW77] J. A. McCall, P. K. Richards, G. F. Walters. Factors in Software Quality. Volume I. Concepts and Definitions of Software Quality. 1977.
http://www.dtic.mil/dtic/tr/fulltext/u2/a049014.pdf

[Neu05] P. G. Neumann. Attaining Robust Open Source Software. In *Perspectives on Free and Open Source Software*. The MIT Press, 2005.

[NL02] D. Noble, M. Letsky. Cognitive-Based Metrics to Evaluate Collaboration Effectiveness. In *Analysis of the Military Effectiveness of Future C2 Concepts and Systems*. RTO-MP-117, NATO Consultation, Command and Control Agency, The Hague, The Netherlands, 23-25 April 2002.
http://www.rto.nato.int/Pubs/rdp.asp?RDP=RTO-MP-117

[Pre00] S. R. Pressman. *Software Engineering - A Practitioner's Approach*. McGraw-Hill International, London, 2000.

[Ray99] E. S. Raymond. *The cathedral and the bazaar*. O'Reilly, 1999.

[RHS06] J. Roberts, I. Hann, S. Slaughter. Understanding the motivations, participation, and performance of open source software developers: a longitudinal study of the Apache projects. *Management Science* 52(7):984–999, 2006.

[SC09] S. A. Shaikh, A. Cerone. Towards a metric for Open Source Software Quality. In *Proceedings of OpenCert 2009*. Volume 20 of Electronic Communications of the EASST. 2009.

[SC10] S. K. Sowe, A. Cerone. Integrating Data from Multiple Repositories to Analyze Patterns of Contribution in FOSS Projects. In *Proceedings of OpenCert 2010*. Volume 33 of Electronic Communications of the EASST. 2010.

[Sch00] F. B. Schneider. Open Source in Security: Visting the Bizarre. In *2000 IEEE Symposium on Security and Privacy*. May 14–17, 2000, Berkley, California, USA, pp. 126–127. IEEE Computer Society, 2000.

[SSS07] S. K. Sowe, I. G. Stamelos, I. M. Samoladas (eds.). *Emerging Free and Open Source Software Practices*. IGI Global, 2007.