Mining Development Repositories To Study the Impact of Collaboration on Software Systems

Nicolas Bettenburg
Software Analysis and Intelligence Lab (SAIL)
Queen's University, School of Computing
Kingston, Ontario K7L 3N6, Canada
nicbet@cs.queensu.ca

ABSTRACT

Software development is a largely collaborative effort, of which the actual encoding of program logic in source code is a relatively small part. Yet, little is known about the impact of collaboration between stakeholders on software quality. We hypothesize that the collaboration between stakeholders during software development has a non-negligible impact on the software system. Information about collaborative activities can be recovered from traces of their communication, which are recorded in the repositories used for the development of the software system. This thesis contributes the following: 1) to make this information accessible for practitioners and researchers, we present approaches to distill communication information from development repositories, and empirically validate our proposed extractors. 2) By linking back the extracted communication data to the parts of the software system under discussion, we are able to empirically study the impact of communication, as a proxy to collaboration between stakeholders, on a software system. Through case studies on a broad spectrum of open-source software projects, we demonstrate the important role of social interactions between stakeholders with respect to the evolution of a software system.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement; D.2.8 [**Software Engineering**]: Metrics; D.2.9 [**Software Engineering**]: Management

General Terms

Management, Measurement, Human Factors

Keywords

Collaboration, Empirical Studies, Software Repositories, Unstructured Data, Socio-Technical Congruence

ESEC/FSE'11, September 5–9, 2011, Szeged, Hungary.

1. INTRODUCTION

Software development as a social activity.
Non-trivial software development is a complex task based on the combined social and technical efforts of software engineers. Modularization makes this complexity manageable [2, 27], and managers use high-level information about the software system to divide the development effort into work teams [15]. This conception is formalized in Conway's Law, who notes in his original work that the source code of a software system stands in direct relation with the organizational structure of the development team, as design decisions require communication among the stakeholders making those decisions [18]. Since software development requires a great amount of social and interpersonal interactions, it comes to no surprise that software engineers typically spend a large fraction of their workday communicating with their co-workers [3]. Lately, research begins to understand that the intricacies of these interactions, such as the forming of social networks and sub-communities [13, 31], work dependencies [16], or daily routines [29] stand in a direct relation to the quality of the final software product [15].

The role of communication in software development.
De Souza et al. provided evidence that communication among software developers is critical for the success of software systems: information hiding led development teams to be unaware of other teams' work, resulting in coordination problems [20]. Bird et al. and Grinter et al. have reported similar findings for distributed software development [11, 22], which highlight communication as the most referenced problem in distributed software development.

Research Hypothesis.
Motivated by the empirical evidence provided in past research, this dissertation proposes to study the impact of collaboration between stakeholders on the quality of a software system more closely. We believe that the quality of a software, is impacted not only by the condition of the source code, but also by the collaborative activities of the stakeholders responsible for the development of the software system. Furthermore, we believe that these collaborative activities also impact the community of developers involved in producing the software system. These believes have led to the formulation of our research hypothesis, which we state as follows:

> *The collaboration between stakeholders impacts the code quality and development community of a software system.*

We conjecture that the communication between stakeholders is a valuable proxy to their collaborative activities. To validate our research hypothesis, we propose the following methodology. First, we develop tools and techniques to extract communication data from development repositories, and empirically validate the proposed approaches through case studies on development repositories provided by open-source software systems. Next, we perform a linking of the extracted communication data back to the parts of the software system referenced within the data. These links enable us to investigate the impact of collaboration between stakeholders on quality and the development community.

The rest of this paper is organized as follows: we present our proposed approach to validating our research hypothesis in Sections 2-4. In each section, we present our motivation, current progress and future research opportunities. In Section 5, we expand on the describing the state of the art beyond the research work presented in Section 1. We conclude our work in Section 6 by outlining the prospective contributions of our work.

2. EXTRACTING COLLABORATION DATA FROM SOFTWARE DEVELOPMENT REPOSITORIES.

Through publicly available data from open-source systems, researchers have access to the rich development repositories of large software systems that are developed by hundreds of developers over extended periods of time. Some of these repositories, such as developer mailing lists, record communication data *explicitly*. At the same time, communication data is also captured *implicitly* in development repositories meant for other tasks, for example developers have been observed to chat through the transaction logs in version control repositories [32].

However, the extraction of communication data from development repositories is a challenging task [4], as communication data was never designed to be pulled out of development repositories. Such data exists mainly in the form of *unstructured data*: arbitrary combinations of natural language text and technical artifacts like source code entities, references to parts of the software system, stack traces, patches, code examples, and other project specific vocabulary. Conventional data and text mining approaches were designed for proper natural language text, as one would find for example in newspaper articles. As such, we cannot readily apply these approaches to mine communication data [8].

Co-organized by Dr. Bram Adams and me, we initiated the 2010 Workshop on Mining Unstructured Data (MUD'10) [4] as a forum for researchers and developers for discussing and sharing the existing work on mining unstructured data into a common reference frame and to discuss the techniques for mining unstructured data.

Within the same line of work presented in the MUD'10 workshop [4], we proposed a lightweight approach based on morphological language analysis to uncover technical artifacts, such as source code examples, stack traces, class and function names, patches, identifiers, and other development specific lingo within unstructured data [5].

We are currently continuing this line of work by extending our empirical validation of the proposed tool. In addition, we are investigating different approaches, including heuristics based on regular expressions, fuzzy text search and vector space models for linking the extracted technical artifacts back to the software system. Through these links we are able to establish a connection between communications and the parts of the software system they talk about.

3. STUDYING THE IMPACT OF COLLABORATION ON THE QUALITY OF A SOFTWARE SYSTEM.

Using the extracted communication data from development repositories, we are able to investigate the impact of collaboration on quality of software systems more closely.

As a first step into this direction, we mined the discussions surrounding the maintenance efforts recorded in the issue trackers of the `Eclipse` and Mozilla `Firefox` software systems. We linked the discussions to the software system through the unique issue report identifiers, which are often referenced by developers when carrying out corrective changes. Through a case study we demonstrated the impact of both, the contents, as well as the dynamics of the communication surrounding corrective maintenance efforts on future software quality [7].

In this line of work, we investigated the relative impact of different aspects of this communication on post-release defects through statistical models. Similar approaches have been successfully used in the past to study the relationship between source code measures and defects [16, 23, 25, 26, 28, 34]. Through our results we observed that the discussions of source code and proposed patches correlate with the risk of future bugs in the areas of source code they reference. Similarly, we discovered that inconsistency in communication flow among stakeholders, for example extended periods of silence, stand in a strong relation with the risk of introducing defects in the software system.

Additionally, we demonstrated that measurements of the communication surrounding corrective maintenance can not only explain post-release defects as well as traditional models based on code metrics, but also that a combination of traditional models based on code metrics, and models based on communication metrics explains post release defects better than each model separately [6].

We are currently investigating the possibilities of increasing the performance of traditional defect prediction methods through the use of collaboration measures.

4. STUDYING THE IMPACT OF COLLABORATION ON THE DEVELOPMENT COMMUNITY.

In the first line of our contributions we propose to analyze the relationships between collaboration and the quality of a software system. However, evidence reported by past research, such as the findings of Bird et al. on the forming of sub-communities [10], led us to strongly believe that collaboration does impact software systems beyond source code: as software development requires a large amount of social and interpersonal interactions, we hypothesize that there is also an impact on the development community responsible for the creation of the software.

Since the quality of a software system is influenced by both, the state of the source code, as well as the people producing the source code, we propose to investigate the

impact of collaboration on the stakeholders responsible for providing source code changes.

Within this line of work, we are currently studying the contribution management processes of several successful open-source systems and the communication surrounding these processes. Open-source systems rely on a thriving community surrounding the software system that carries out much of the adaptive development effort [9]. As software systems are constantly changing [24], the management of new source code, contributed by the development community takes on important role in the development of open-source software. Contribution management requires communication between a variety of different stakeholders, including the discussion of the design and implementation of contributions, peer-reviews carried out for means of software quality assurance, and the integration of the proposed changes into future releases. We conjecture that irregularities in the communication surrounding these processes are potentially disruptive to the integration of future changes into the software. For example, a member of that development community who does not get timely feedback on his proposed source code changes in a timely fashion might get discouraged to contribute to the software system in the future and leave the community.

Our preliminary findings suggest the existence of several strong relationships between the way an open-source software community collaborates and the evolution of their organizational structure.

5. STATE OF THE ART

Beyond the research work presented in Section 1, we recognize the following lines of research that are closely related to the work presented in this thesis.

Socio-Technical Congruence.
Literature documents a variety of factors leading to communication problems in software development, such as frequency [3], the number of stakeholders involved [14, 17], the non-sharing of knowledge [19], and architectural modularity [21]. Amrit et al. were the first to propose a formalization of developer communication in so-called "affiliation networks" [1], within the context of validating Conway's law. This idea has been picked up by Valetto et al., who examine socio-technical networks, which merge social networks mined from communication meta-data with collaboration networks mined from source code [30]. In this line of work, Valetto et al. propose the notion of socio-technical congruence as a degree of fit between communication and coordination of developers related to the same software component [30]. Cataldo et al. studied the relation between socio-technical congruence and the software development process [15, 17].

Use of collaboration metrics in defect prediction.
Research has since picked up social networks and social network analysis to describe developer collaboration. To date, most research effort is based on both social network analysis and socio-technical networks. Some notable instances include the work by Bird et al., who use social network analysis on social networks mined from email repositories [10,13]. These social networks lay the foundation for the use of socio-technical networks to predict software defects [12]. Similarly, Zimmermann et al. use social network measures on dependency graphs to predict defects [33], and Wolf et al. present a case study on the use of social network analysis measures obtained from inter-developer communication in the IBM Jazz repository to predict build failures [31].

6. CONCLUSIONS AND CONTRIBUTIONS

Traces of collaborative software development activities can be found in the repositories, developers use on a day to day basis, such as version control systems, issue tracking systems and email discussion lists. By extracting communication data from development repositories and linking them back to the software system, we are able to investigate the impact of collaboration on the quality and the development community of a software system.
The *technical contributions* of our work center around the development and empirical validation of methods to extract communication information from development repositories. In particular, we propose the following contributions:

- **Tools and techniques for mining communication information from development repositories.** As a first step towards making communication data hidden in development repositories accessible for researchers and practitioners, this dissertation presents tools and methods to distill communication information from development repositories, such as mailing list archives, version control systems, or issue tracking databases.

- **Empirical validation of the applicability of the presented tools and heuristics.** We demonstrate the applicability of these tools by empirically validating the proposed methods and heuristics through case studies on a wide range of development repositories of different software systems. We propose tools and techniques for the automated extraction of communication data from software development repositories.

The *conceptual contributions* of our work center around using the extracted communication information to investigate the impact of collaboration on software quality. In particular, we propose the following contributions:

- **Empirical validation of the impact of collaboration on the quality of a software system.** Through an empirical study on the impact of communication surrounding the software development process and the quality of a software system we uncovered several factors influencing the risk of post-release defects.

- **Empirical validation of the impact of collaboration on the development community.** Through an empirical study on the communication surrounding the process of contribution management in open source software systems, we demonstrate the impact of collaboration on the community developing the software system, which in return effects the integration of adaptive, corrective and prescriptive changes into the software.

7. REFERENCES

[1] C. Amrit, J. Hillegersberg, and K. Kumar. A social network perspective of conway's law. In *Proceedings of the CSCW Workshop on Social Networks*, Chicago, IL, 2004.

[2] C. Y. Baldwin and K. B. Clark. *Design Rules: The Power of Modularity Volume 1*. MIT Press, Cambridge, MA, USA, 1999.

[3] A. Begel, Y. P. Khoo, and T. Zimmermann. Codebook: discovering and exploiting relationships in software repositories. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ICSE '10, pages 125–134, New York, NY, USA, 2010. ACM.

[4] N. Bettenburg and B. Adams. Workshop on Mining Unstructured Data (MUD), because "mining unstructured data is like fishing in muddy waters"! In *Proceedings of the 17th IEEE Working Conference on Reverse Engineering*, pages 277 –278. IEEE Computer, 2010.

[5] N. Bettenburg, B. Adams, A. E. Hassan, and M. Smidt. A lightweight approach to uncover technical artifacts in unstructured data. In *ICPC'11: Proceedings of the 19th IEEE International Conference on Program Comprehension*, page to appear, Kingston, Ontario, Canada, 2011. IEEE Computer.

[6] N. Bettenburg and A. E. Hassan. Studying the impact of social structures on software quality. In *ICPC'10: Proceedings of the 18th IEEE International Conference on Program Comprehension*, pages 124–133, Braga, Portugal, 2010. IEEE.

[7] N. Bettenburg and A. E. Hassan. Studying the impact of social structures on software quality. *submitted to the Editorial Office of Empirical Software Engineering*, 2011.

[8] N. Bettenburg, E. Shihab, and A. E. Hassan. An empirical study on the risks of using off-the-shelf techniques for processing mailing list data. In *Proceedings of the 25th IEEE International Conference on Software Maintenance*, Edmonton, Alberta, Canada, 2009. IEEE Computer.

[9] C. Bird, A. Gourley, and P. Devanbu. Detecting Patch Submission and Acceptance in OSS Projects. In *Proceedings of the Fourth International Workshop on Mining Software Repositories*. IEEE Computer Society, 2007.

[10] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan. Mining Email Social Networks in Postgres. In *Proceedings of the Third International Workshop on Mining software repositories (Challenge Track)*. ACM, 2006.

[11] C. Bird, N. Nagappan, P. Devanbu, H. Gall, and B. Murphy. Does Distributed Development Affect Software Quality? An Empirical Case Study of Windows Vista. *Communications of the ACM*, 52(8):85–93, August 2009.

[12] C. Bird, N. Nagappan, P. Devanbu, H. Gall, and B. Murphy. Putting it All Together: Using Socio-Technical Networks to Predict Failures. In *Proceedings of the 17th International Symposium on Software Reliability Engineering*, 2009.

[13] C. Bird, D. Pattison, R. D'Souza, V. Filkov, and P. Devanbu. Latent social structure in open source projects. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, SIGSOFT '08/FSE-16, pages 24–35, New York, NY, USA, 2008. ACM.

[14] F. P. Brooks, Jr. *The mythical man-month (anniversary ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[15] M. Cataldo, J. D. Herbsleb, and K. M. Carley. Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity. In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, ESEM '08, pages 2–11, New York, NY, USA, 2008. ACM.

[16] M. Cataldo, A. Mockus, J. A. Roberts, and J. D. Herbsleb. Software dependencies, work dependencies, and their impact on failures. *IEEE Transactions on Software Engineering*, 35(6):864–878, 2009.

[17] M. Cataldo, P. A. Wagstrom, J. D. Herbsleb, and K. M. Carley. Identification of coordination requirements: implications for the design of collaboration and awareness tools. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, CSCW '06, pages 353–362, New York, NY, USA, 2006. ACM.

[18] M. Conway. How do committees invent. *Datamation*, 14(4):28–31, 1968.

[19] B. Curtis, H. Krasner, and N. Iscoe. A field study of the software design process for large systems. *Commun. ACM*, 31:1268–1287, November 1988.

[20] C. R. B. de Souza, D. Redmiles, L.-T. Cheng, D. Millen, and J. Patterson. How a good software practice thwarts collaboration: the multiple roles of apis in software development. *SIGSOFT Softw. Eng. Notes*, 29(6):221–230, October 2004.

[21] C. R. B. de Souza, D. Redmiles, and P. Dourish. "breaking the code", moving between private and public work in collaborative software development. In *Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work*, GROUP '03, pages 105–114, New York, NY, USA, 2003. ACM.

[22] R. E. Grinter, J. D. Herbsleb, and D. E. Perry. The geography of coordination: dealing with distance in r&d work. In *Proceedings of the international ACM SIGGROUP conference on Supporting group work*, GROUP '99, pages 306–315, New York, NY, USA, 1999. ACM.

[23] A. E. Hassan. Predicting faults using the complexity of code changes. In *ICSE '09: Proceedings of the 31st International Conference on Software Engineering*, pages 78–88. IEEE Computer Society, 2009.

[24] M. M. Lehman. On understanding laws, evolution, and conservation in the large-program life cycle. *Journal of Systems and Software*, 1:213–221, 1980.

[25] N. Nagappan and T. Ball. Use of relative code churn measures to predict system defect density. In *ICSE '05: Proceedings of the 27th International Conference on Software Engineering*, pages 284–292. ACM, 2005.

[26] N. Nagappan and T. Ball. Using software dependencies and churn metrics to predict field failures: An empirical case study. In *ESEM '07: Proceedings of the First International Symposium on Empirical Software Engineering and Measurement*, pages 364–373. IEEE Computer Society, 2007.

[27] D. L. Parnas. On the criteria to be used in decomposing systems into modules. *Commun. ACM*, 15:1053–1058, December 1972.

[28] A. Schröter, T. Zimmermann, and A. Zeller. Predicting component failures at design time. In *ISESE '06: Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering*, pages 18–27. ACM, 2006.

[29] J. Śliwerski, T. Zimmermann, and A. Zeller. When do changes induce fixes? In *MSR '05: Proceedings of the 2005 International Workshop on Mining Software Repositories*, pages 1–5. ACM, 2005.

[30] G. Valetto, M. Helander, K. Ehrlich, S. Chulani, M. Wegman, and C. Williams. Using software repositories to investigate socio-technical congruence in development projects. In *Proceedings of the Fourth International Workshop on Mining Software Repositories*, MSR '07, pages 25–, Washington, DC, USA, 2007. IEEE Computer Society.

[31] T. Wolf, A. Schröter, D. Damian, and T. Nguyen. Predicting build failures using social network analysis on developer communication. In *ICSE '09: Proceedings of the 31st International Conference on Software Engineering*, pages 1–11. IEEE Computer Society, 2009.

[32] A. T. T. Ying, J. L. Wright, and S. Abrams. Source code that talks: an exploration of eclipse task comments and their implication to repository mining. In *Proceedings of the 2005 international workshop on Mining software repositories*, MSR '05, pages 1–5, New York, NY, USA, 2005. ACM.

[33] T. Zimmermann and N. Nagappan. Predicting defects using network analysis on dependency graphs. In *ICSE '08: Proceedings of the 30th international conference on Software engineering*, pages 531–540. ACM, 2008.

[34] T. Zimmermann, R. Premraj, and A. Zeller. Predicting defects for eclipse. In *Proceedings of the Third International Workshop on Predictor Models in Software Engineering*, May 2007.