A Framework for Collaborative Software Development Analytics

Eleni Stroulia
Computing Science Department
University of Alberta
Edmonton, AB T6G 2E8
Canada
stroulia@ualberta.ca

Fabio Rocha
Computing Science Department
University of Alberta
Edmonton, AB T6G 2E8
Canada
fabiorocha@ualberta.ca

Nikolaos Tsantalis
Computing Science Department
University of Alberta
Edmonton, AB T6G 2E8
Canada
tsantalis@ualberta.ca

ABSTRACT
In this position paper we describe a conceptual model for analyzing collaborative software development for the purposes of (a) providing to the developers information which can effectively increase their awareness of their project status and support their development tasks, and (b) supporting its more effective management.

Author Keywords
Collaborative software development; Analytics; Software cost

ACM Classification Keywords
D.2.6 Programming Environments: Integrated environments
D.2.9 Management: Cost estimation
D.2.9 Management: Life cycle
D.2.9 Management: Productivity
D.2.9 Management: Programming teams

General Terms
Management; Human Factors; Economics

INTRODUCTION AND BACKGROUND
We are noticing two interesting trends relevant to collaborative software development (CSD). On one hand, the software community recognizes the need for a new generation of IDEs designed to flexibly support collaborative and, to a large extent, distributed software development. On the other, we are observing the increased adoption of web2.0 communication tools by software developers, who communicate within their teams and with the community at large through a variety of informal channels on matters relevant to their tasks. We believe that this "tension" poses a substantial challenge (and, at the same time, offers an exciting opportunity) to conceptualizing the tool support for CSD in the future.

In our group, we have been long working on *analyzing the artifacts* involved in CSD, for the purpose of extracting higher-level knowledge that would help *increase the team's awareness* of the project. In this task, we had two methodological objectives: (i) to be inclusive (by considering data produced by the variety of tools used by the team); and (ii) to link the collected data through a variety of technologies.

To that end, in the context of the WikiDev project, we have worked on two major tasks with the objective of developing a lightweight, wiki-based CSD platform. First, we developed a suite of APIs for ingesting data into a common MediaWiki-based repository and corresponding adapters for external systems to export this data through the above APIs [1]. Second, we developed a suite of analyses for extracting explicit and implicit relations on the basis of which to link the collected data, including references to common software artifacts and team members, as well as implicit relations between developers and their tasks, captured in natural-language text used in informal communications [2].

There have been several similar projects aiming to collect and cross-reference data[1], from independent tools, about software products and processes. For example, DrProject [3] is a web-based software project management portal that integrates a revision history viewer with issue tracking, mailing list management, a wiki, and other features. Atlassian JIRA[2] is a project-tracking tool for bugs/defects that allows to link issues to related source code, plan agile development, monitor activity and report on project status. Finally, Hackystat[3] is an open source framework for collection and analysis of software development process and product data, where the users (developers) have to install sensors (plug-ins) to their development tools (e.g.,

[1] Due to the restrictions on the length of this position paper, we do not discuss here a number of relevant projects focusing on repository analyses. Instead, we choose to eclectically mention only a few whose objectives were primarily to serve as integration middleware among a variety of tools.
[2] http://www.atlassian.com/software/jira/overview
[3] http://code.google.com/p/hackystat/

CVS, eclipse, Atlassian JIRA). The sensors collect data and send it to a centralized repository (accessible through web-services) to form higher-level abstractions of the data. Hackystat's analyses mainly concern metrics such as test coverage, complexity, coupling, commit frequency and size.

More recently we have started to work with Jazz RTC [4], a comprehensive CSD tool that offers, on one hand, seamless integration with Eclipse, and, on the other, a web-based user interface for integration with other tools. In this context, the data-collection challenge is, to a great extent, resolved, but the problem of a CSD linked-data framework is still relevant, and in fact a core task in the future research CSD agenda. In this position paper, we put forward a general conceptual framework for analyzing CSD for the dual purpose of providing to the developers information which can effectively increase their awareness of their project status and support their development tasks, and of supporting its more effective management. This latter objective, we believe, is essential for understanding the nature of costs associated with CSD, and the corresponding opportunities for value creation.

A PROPOSAL FOR A CSD ANALYTICS FRAMEWORK

The most mature proposal to date for a "software analytics framework" has been "the 46 questions" that developers ask, empirically identified by Fritz and Murphy [5].

Our proposal, discussed in this position paper, is developed from a theoretical (not an empirical) perspective and aims to make "collaboration" a first-class dimension of analysis. At the same time, given how comprehensive the above framework is, we have tried to classify the 46 questions above in terms of our proposal in order to enable the seamless migration of any work framed around these questions in our framework.

Our framework proposes the analysis of CSD in terms of (a) the people involved, (b) the products of their work, and (c) the process they follow, similar to the PCANS model, proposed by Krackhardt and Carley [6] to represent the structure of an organization through the different relationships among three domain elements (individuals, tasks and resources). Following this concept, we have conceived a comprehensive set of analyses, each of which is viewed as linking data within or across these dimensions. In particular, we are interested in analyses that focus on cost-value questions.

For example, considering *People vs. People analyses*, we are interested in comparing the activity of selected team members over time, and in aggregating related activities within a team. In the context of such analyses, the other dimensions (products and process) become analysis facets: for example, in aggregating related activities, we may want to distinguish these activities in terms of the (types of) products they affect (design models vs. production code vs. test code vs. communication artifacts) or in terms of the process steps in which they occur.

In the context of *People vs. Products analyses*, we wish to recognize the role of individuals with respect to specific software artifacts (designer, owner developer, and tinkerer) and the amount and the quality of the overall contribution of an individual to the project as a whole.

In the context of *People vs. Process analyses*, we wish to examine the activities of individuals, and the role they play, in the project lifecycle, the extent to which these roles conform with the "official" process model adopted by the project, and the relative information content of the process-specific communication and coordination artifacts (like "stories" in agile development or work-items in RUP as implemented in Jazz) vs. the variety of informal communication channels increasingly adopted by developers like automatic status updates [7].

In the *Products vs. Products analyses*, the relations among software artifacts become the subject of examination. These may be "reflection" relations, where implementation-level artifacts are compared against higher-level (requirements- and design-level) artifacts for the purpose of establishing (non-)conformance or traceability links or assessing progress. When products of the same type are examined, dependency relations are identified, whether direct and explicit (such as "calls" or "uses") or hidden and implicit (such as "co-evolves" [8]). Adopting a *People* facet in these analyses, we can examine the properties of the social networks implied by the relations among products upon the developers working with these products.

Under the umbrella of *Products vs. Process analyses*, we see investigations of how the number and types of products manipulated by the team, as well as the amount and types of changes they suffer, evolves over time in the course of the project. Analysis aimed at recognizing whether a phase is fundamentally one of feature expansion or one of refactoring [9], [10]. Again adopting a *People* facet, we can examine the relative amount and type of contributions the various team members make.

Finally, under the *Process vs. Process analyses*, we are interested in analyzing the conformance of the team's activities to their officially adopted process, based on how they coordinate these activities through formal and informal communication channels. The use of informal communication channels is a relatively recent phenomenon, and we anticipate a variety of uses for them in support of more traditional processes.

The Cost Dimension

We believe that the above framework can be quite useful in organizing the various CSD analyses that we envision will be the subject of research in the near- and medium-term future. However, the types of analyses that, we believe, will become particularly relevant in the context of CSD, are those examining issues of cost and value, in ways that may lead to management decisions.

For example, *People vs. People* communication analyses may focus on estimating organizational costs and supporting decisions on how to distribute a team physically or how to adapt the process to better support and simplify communication and coordination. *People vs. Products* analyses for monitoring productivity may also drive estimates of organizational costs and decisions on how to assign team members to working on specific products. Finally, *Products vs. Products* analyses for monitoring software quality can drive software cost estimates and guide maintenance vs. redevelopment decisions.

SUMMARY

Future CSD will be quite varied in terms of the numbers of people involved in a project at any point in time, the types of software and coordination/communication artifacts manipulated by the team members, and the process models adopted by these teams and their variations. The systematic analysis of CSD, especially for cost-value trade-off analysis and decision making, will benefit from a systematic framework such as the one we propose here.

ABOUT THE AUTHORS

Eleni Stroulia has been interested in software-development analyses, both from a process and from a product perspective. Starting with JDEvAn [11] and CVSChecker [12], following with WikiDev and WebDiff [13] and more recently with Jazz, she is interested in supporting the management of the software-development process based on information extracted from the project's history.

Fabio Rocha is a Master's student at the University of Alberta. His research interests include collaborative software systems and business concerns of software development. He is currently working on the development of analytics services and visualizations for the IBM Jazz collaborative software delivery platform.

Nikolaos Tsantalis is a Postdoctoral Fellow at the Department of Computing Science, University of Alberta, Canada. His research interests include design pattern mining, identification of refactoring opportunities, and design evolution analysis. He is currently working on the development of differencing services for heterogeneous software artifacts and their integration in Web2.0 systems.

ACKNOWLEDGEMENTS

This work was funded by NSERC, AITF (Alberta Innovates Technology Futures) and IBM. The authors are also grateful to Marios Fokaefs, Diego Serrano, Maryam Hasan and Ken Bauer for their contributions to the work that led to these ideas.

REFERENCES

1. Bauer, K., Fokaefs, M., Tansey, B. and Stroulia, E. WikiDev 2.0: Discovering Clusters of Related Team Artifacts. *In Proc. of the 2009 Conference of the Center for Advanced Studies on Collaborative Research* (CASCON '09), 174-187.
2. Hasan, M., Stroulia, E., Barbosa, D. and Alalfi, M. Analyzing natural-language artifacts of the software process. *In Proc. of the 2010 IEEE International Conference on Software Maintenance* (ICSM '10), 1-5.
3. Reid, K.L. and Wilson, G.V. DrProject: A Software Project Management Portal to Meet Educational Needs. *In Proc. of the 38th SIGCSE Technical Symposium on Computer Science Education* (SIGCSE '07), 317-321.
4. Cheng, L.-T., Hupfer, S., Ross, S. and Patterson, J. Jazzing up Eclipse with Collaborative Tools. *In Proc. of the 2003 OOPSLA workshop on eclipse technology eXchange* (eclipse '03), 45-49.
5. Fritz, T. and Murphy G.C. Using Information Fragments to Answer the Questions Developers Ask. *In Proc. of the 32nd ACM/IEEE International Conference on Software Engineering* (ICSE '10), vol. 1, 175-184.
6. Krackhardt D. and Carley K. M. A PCANS Model of Structure in Organizations. *In International Symposium on Command and Control Research and Technology*, 1998, 113-119.
7. King, A. and Lyons, K. Automatic Status Updates in Distributed Software Development. *In Proc. of the 2nd International Workshop on Web 2.0 for Software Engineering* (Web2SE '11), 19-24.
8. Xing, Z. and Stroulia, E. Understanding the Evolution and Co-evolution of Classes in Object-oriented Systems. *International Journal of Software Engineering and Knowledge Engineering*, vol. 16, 1 (2006), 23-52.
9. Schofield, C., Tansey, B., Xing, Z. and Stroulia, E. Digging the Development Dust for Refactorings. *In Proc. of the 14th IEEE International Conference on Program Comprehension* (ICPC '06), 23-34.
10. Xing, Z. and Stroulia, E. Analyzing the Evolutionary History of the Logical Design of Object-Oriented Software. *IEEE Transactions on Software Engineering*, vol. 31, 10 (2005), 850-868.
11. Xing, Z. and Stroulia, E. The JDEvAn Tool Suite in Support of Object-Oriented Evolutionary Development. *In Companion of the 30th International Conference on Software Engineering* (ICSE Companion '08), 951-952.
12. Liu, Y., Stroulia, E. and Erdogmus, H. Understanding the Open-Source Software Development Process: a Case Study with CVSChecker. *In Proc. of the First International Conference on Open Source Systems* (OSS '05), 154-161.
13. Tsantalis, N., Negara, N. and Stroulia, E. WebDiff: A Generic Differencing Service for Software Artifacts. *In Proc. of the 27th IEEE International Conference on Software Maintenance* (ICSM '11), 586-589.