Chat to Avoid Broken Builds

Jazz Innovation Award Proposal

Daniela Damian and Adrian Schröter
Software Engineering Global interAction Lab
University of Victoria
PO Box 3055, STN CSC
British Columbia V8W 2P6, Canada
danielad@cs.uvic.ca, schadr@uvic.ca

1. MOTIVATION

As software systems grow larger over time, it becomes more unlikely that a single person can keep the overview over such a system. Often it is challenging enough to keep track of the component one is working on. Changes made by team members within the component you are working on or changes to API's you use can cause problems if they are not communicated properly. With a growing component, a growing team, and a growing usage of API's such miss communications can occur more frequently and cause severe problems, such as broken builds or bugs in software release.

IBM's Jazz platform together with the Eclipse based Rational Team Concert Client supports team collaboration and integrates source code control with a communication platform. This integration enables us to develop and implement the recommender system *"Chat to Succeed"* that recommends developers to whom to talk to about what in order to increase software quality. Additionally, it will support managers to spot malicious patterns that need to be broken and best practices that should be fostered.

2. CHAT TO SUCCEED

This proposal builds on our current Jazz-related research that examined integration and communication practices in Jazz and identified a positive and strong relationship between broken builds and the quality of the communication of the team involved in the build [9]. In this project we plan to take it one step further by recommending what actions need to be taken in order to prevent a build from braking. We tackle this goal by providing developers and managers alike with recommendations to improve collaboration within a team. Therefore we formulate the following two objectives.

2.1 Objectives

The goal of this project is to develop and evaluate a recommender system that helps development teams to avoid broken integration builds by:

- Supporting developer collaboration by recommending people to talk to and topics to talk about in order to prevent the next build from braking.
- Supporting effective management through recommendations for managers to identify emergent best collaboration practices, or malicious practices that, for instance, often lead to broken builds.

Reaching those objectives enables developers and managers to adjust the team collaboration dynamically, both at the individual and group level, consequently increasing their development performance by lowering the number of broken builds.

2.2 Development Plan

To fulfill these objectives we develop a recommender system that uses project historical information on developer communication, integration build outcomes, and source code changes to recommend ways to prevent the current build from braking. Our recommender system will include a preparation (Step 1 and 2) and a recommendation (Step 2) component:

1. We extract developer relationships and examine them in relation to the health of the past integration builds. Specifically, we create the communication and technical dependency networks for each past build in the project. A *communication network* for a build contains all people that commented on work items that are linked to that build. A communication line between two developers exists if they commented on at least one shared work item. Then we build the *technical dependency network* that contains all people that made source code changes in the project. Each developer that changes a part of source code is connected to everyone that uses that part of the source code.
2. Having identified the communication and technical dependency networks for each build, we examine each pair of developers in the two networks. We identify whether the developers share a technical dependency, communication dependency, both, or none, and annotate the pair of developers and their relationship with the respective build outcome.
3. To make recommendations for the upcoming build we first construct the respective communication and technical networks and develop a recommendation algorithm that considers the properties of these networks in relation to the knowledge gained at Step 1. For example we could, for each developer pair, count how often their current relationship (Do they share a technical dependency, communication dependency, both, or none?) occurs with broken and successful builds. If the ratio between the occurring in broken and successful builds is higher than the ratio between broken and successful builds, then we make the recommendation to change the developer relationship with respect to communication. This means if those developers did not communicate with each other we recommend to do so, in case they did we recommend to reevaluate the gained information.

To managers, Step 1 is of particular interest, because it identifies emergent best collaboration practices, such as certain communication structures, or malicious patterns those that are likely to lead to broken builds.

To further maximize the efficiency of the recommender system, we integrate our earlier developed red build predictive model [9]. This gives developers and managers an indicator when it is most crucial to pay attention to our recommendation in order to prevent the next build from braking.

2.3 Project Plan

We organize the project plan along four major ***milestones*** in the following order:

1. Developing a recommendation algorithm and automate the extraction of communication and techical networks as in Step 1.
2. Implement a prototype as RTC/Eclipse plug-in.
3. Conduct a field study to evaluate the prototype.
4. Release version 1.0 of the plug-in.

2.4 Resources

The resources needed to the fulfill this project include: a full-time PhD student to conduct the research and develop the recommender algorithm, a programmer to assist the PhD student in the RTC plug-in, and access to a development team that uses Jazz in their development to evaluate the recommender system. This leads us to appling for the full amount of 25,000$.

3. RESEARCH CONTRIBUTION

Chat to Succeed expands the knowledge around socio-technical congruence, which studies the alignment of technical and social dependencies of a development team. Research questions in this area are, for instance, do all developers communicate their changes to others that depend on the changed software part? We have obtained a first positive feedback on this project idea recently [7].

Similar to other studies of socio-technical congruence, we analyze this congruence but are not as concerned with identifying all discrepancies between the social and technical networks in the project, but those discrepancies that specifically lead to broken builds.

4. VALUE TO THE JAZZ COMMUNITY

The Jazz community is provided with a RTC plug-in that enables developers and managers to improve their teams collaboration. This improvement is always towards a specific goal, such as preventing the next build from failing. Our recomemnder system will minimize the time spent with fixing broken builds and allow a more fluent development process in Jazz.

5. DISSEMINATION AND OPENNESS

We plan to disseminate our results by announcing them on jazz.net as well as publish papers and articles in relevant software engineering conferences, workshops, and journals, such as the International Conference on Software Engineering(ICSE), Foundations of Software Engineering (FSE), Computer Supported Collaborative Work (CSCW), International Conference on Global Software Engineering (ICGSE), or Eclipse Technology eXchange (ETX).

Additionally we plan to develop our prototypes and software in the soon available Jazz Research Community repository to gather additional early feedback from the research community. In addition each prototype will be made publicly available after completion on our website and announced on jazz.net.

6. TRACK RECORD/BACKROUND

Our research group has extensive experience with Jazz development and Jazz-related research. We were among the first research group to receive the first Jazz Grand Awards in 2005. Since then we became an active part of the Jazz research community and lately advanced to take a leading role. Together with Li-Te Cheng and Gail Murphy we are organizing the first workshop on infrastructure for research in collaborative software engineering [1] at FSE 2008 venue that attracted mostly Jazz related publications.

We also implemented two prototypes as Eclipse plug-ins leveraging the Jazz repository, *FATE* and *Related Contributors* [5, 6]. Both tools are meant to increase the awareness about, in case of *FATE*, related artifacts, such as work items, and, in case of *Related Contributors*, artifact related developers. These tools have been show-cased at OOPSLA 2007 at the Jazz Birds of a Feather session.

Furthermore, we published reports on two projects that exmined communication in the Jazz team. The first project investigated the communication delay on work item discussions across different sites [4] and received an invitation for journal publication. The second investigated the relation between communication behavior and broken builds [9]. The project, called "The Red Build Predictive Model" was also presented at the IBM Rational Developer Conference 2008 at the Jazz Main Event.

Apart from these two project we gathered experience in extracting [10] and analyzing social networks [2, 3] that arose around a single or a set of tasks. Our experience with developing plug-ins, extracting and analyzing social networks, as well as previous experience in mining software repositories [8] will enable us a successful implementation of this proposed project.

7. REFERENCES

[1] L.-T. Cheng, D. Damian, G. Murphy, and A. Schröter. iReCoSE 2008 - 1st International Workshop on Integrated Support or Integrated Overhead? Infrastructure for Research in Collaborative Software Engineering. In *Proccedings of the16th ACM SIGSOFT International Symposium on the Foundation of Software Engineering*, New York, NY, USA, 2008. ACM.

[2] D. Damian, S. Marczak, and I. Kwan. Collaboration patterns and the impact of distance on awareness in requirements-centred social networks. *15th IEEE International Conference on Requirements Engineering (RE 2007)*, pages 59–68, 15-19 Oct. 2007.

[3] S. Marczak, D. Damian, U. Stege, and A. Schröter. Information brokers in requirement-dependent social networks. In *16th IEEE International Conference on Requirements Engineering (RE 2008)*, September 8–12 2008.

[4] T. Nguyen, T. Wolf, and D. Damian. Global software development and delay: Does distance still matter? In *Proceedings of the 3rd International Conference on Global Software Engineering (ICGSE 2008)*. IEEE Computer Society, August 2008.

[5] L. D. Panjer, D. Damian, and M.-A. Storey. Cooperation and coordination concerns in a distributed software development project. In *CHASE '08: Proceedings of the 2008 international workshop on Cooperative and human aspects of software engineering*, pages 77–80, New York, NY, USA, 2008. ACM.

[6] L. D. Panjer and I. Kwan. Jazz Team Communication Tools. In *Jazz Birds of a Feather at OOPSLA (OOPSLA Jazz BoF 2007)*, 2007.

[7] A. Schröter, I. Kwan, L. D. Panjer, and D. Damian. Chat to Succeed. In *Proceedings of the 1st Workshop on Recommender Systems in Software Engineering (RSSE 2008)*, New York, NY, USA, November 2008. ACM.

[8] A. Schröter, T. Zimmermann, R. Premraj, and A. Zeller. If your bug database could talk... (short paper). In *Proceedings of the 5th International Symposium on Empirical Software Engineering. Volume II: Short Papers and Posters*, pages 18–20, September 2006.

[9] T. Wolf, A. Schröter, D. Damian, and T. Nguyen. Communication, Coordination and Integration. Technical Report DCS-322-IR, University of Victoria, June 2008.

[10] T. Wolf, A. Schröter, D. Damian, T. Nguyen, and L. D. Panjer. Mining Task-based Social Networks to Explore Collaboration in Software Teams. *IEEE Software Special Issue on Mining s Repositories, to Appear*, 2009.