Profile-based View Composition in Development Dashboards

Martin Brandtner, Philipp Leitner, Harald Gall

University of Zurich, Switzerland
{brandtner, leitner, gall}@ifi.uzh.ch

Abstract

Continuous Integration (CI) environments cope with the repeated integration of source code changes and provide rapid feedback about the status of a software project. However, as the integration cycles become shorter, the amount of data increases, and the effort to find information in statically composed CI dashboards becomes substantial. We want to address the shortcoming of static views with a so-called Software Quality Assessment (SQA) mashup and profiles. *SQA-Mashup* describes an approach to enable the integration and presentation of data generated in CI environments. In addition, *SQA-Profiles* describe sets of rules to enable a dynamic composition of CI dashboards based on stakeholder activities in tools of a CI environment (e.g., version control system).

1 Introduction

A fundamental aspect of Continuous Integration (CI) according to Fowler is visibility: "*[...] you want to ensure that everyone can easily see the state of the system and the changes that have been made to it.*"[4]. The integration of a modern CI environment within the development process of a software project is fully automated, and its execution is triggered after every commit. With each integration run, modern CI tools generate a bulk of data. However, this data is scattered across the entire CI environment and analyzing it, for instance, to monitor quality of a system, is a time consuming task. This can delay the rapid feedback cycles of CI and one of its major benefits is then not realized.

The need for an integration and the tailoring of data generated by the tools used in a CI environment is expressed in studies that address the information needs of software developers. Questions are, for example, *What has changed between two builds [and] who has changed it?* [5].

In this work, we briefly revisit our approaches to support the answering of such questions. We cherry-picked parts of our previous research, which was published at the International Conference on Software Analysis, Evolution, and Reengineering [2] and the Information and Software Technology Journal [1].

2 SQA-Mashup

The goal of the SQA-Mashup approach [1] is to integrate the information scattered across CI tools into a single view. The information is presented according to the information needs of different stakeholders.

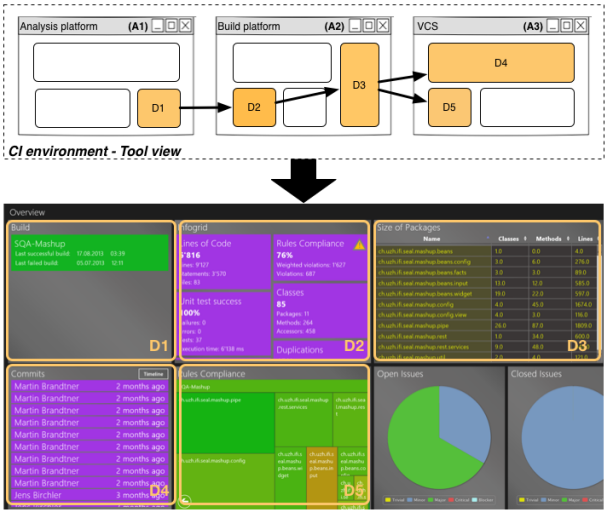

Figure 1: SQA-Mashup: Integrated view

We aim for a fast and easy way to analyze and communicate the actual state, the current health, and the most recent changes of a system. Figure 1 depicts the integration of data from a CI environment to present it to different stakeholders. The *CI environment – Tool view* in Figure 1 represents the information gathering process of, for example, a software tester during a quality measurement review. Starting from a quality analysis platform (*A1*), where the tester detects an increase of convention violations, the developer navigates to the build platform (*A2*) and from there to the associated source code commits (*A3*) to locate the causes of the violations. However, during a development or testing task, such as bug fixing, only some data of each CI tool is relevant to solve the task. The highlighted elements (*D1-D5*) in the single tools indicate the data accessed by the software tester during his review. Our proposed mashup integrates the single pieces of data from different CI tools into a condensed model to dynamically represent the data according to the information needs of a stakeholder.

To evaluate our model-based approach, we de-

signed and conducted a controlled user study with 16 participants who had to solve nine software maintenance tasks. The control group had to use state-of-the-art CI tools and a social coding platform: Jenkins-CI and SonarQube as the CI tools and GitHub as the social coding platform. The experimental group had to use SQA-Mashup, which is a tool that was implemented according to the equally named model.

Overall, we found evidence that the participants of the experimental group solved the tasks of our study faster (57%) and with a higher correctness (21.6%). When analyzing these differences on the level of the individual tasks we found major insights in the benefits of monitoring the CI process and the capabilities of existing CI tools. On the one hand, regarding single aspects of software quality, such project size or code complexity, existing CI-tools provide already good support. On the other hand, we found evidence that monitoring software quality during CI can substantially benefit from an integrated view.

3 SQA-Profiles

The aim of the SQA-Profiles approach [2] is the profiling of stakeholders within a group of software project stakeholders. An example for such a stakeholder group is the project management committee (PMC) in Apache projects.[1] In comparison to other groups, such as committers, the covered spectrum of tasks is broader in a PMC. For example, committers work on issues and contribute source code changes. PMC members actively contribute issues and source code as well, but the PMC is additionally in charge of project and community management. The management of the project incorporates tasks, such as monitoring or gatekeeping. A benefit of using a committee compared to a single manager is the ability to share tasks among the different committee members. For example, some PMC members might focus on source code integration, whereas others taking care of the issue management and gatekeeping. However, the resulting different focus of the PMC members requires a different view on the data presented in dashboards as well [3]. With SQA-Profiles we tried to extract the different focus of PMC members from a collection of Apache projects and to describe them in a project-independent and rule-based manner. We extracted following activity attributes: commits, merges, issue status changes, issue comments, issue assignee changes, and issue priority changes. We found four distinct profiles. One of these profiles is the so-called *Integrator* profile. Any PMC member with this profile has a high merge and commit activity in the according Apache project. The thresholds for a high, medium, and low activity in a certain attribute get computed automatically and individually for each project.

To evaluate our proposed approach, we studied the activity data of PMC members from 20 Apache projects that use Java as main language between September 2013 and September 2014.

Overall, we found evidence that activity data mined from the VCS and the issue tracking platform can reflect the tasks of stakeholders within a certain group. The evaluation results (precision: 0.92, recall: 0.78) showed that the automatic SQA-Profile approach performs almost as good as the semi-automatic baseline, which requires project-dependent threshold parametrization. Additionally, we were able to show that an assigned role in a software project does not necessarily reflect the actual activities of a stakeholder. For example, in some projects normal contributors take care of PMC tasks.

[1] `http://www.apache.org/foundation/how-it-works.html`

4 Bridge the Gap

SQA-Mashup and SQA-Profiles address different aspects to bridge the gap between the information available and the information needed by a stakeholder. The first provides a model to integrate and present CI data and the second enables a categorization of stakeholders based on their activity. To finally bridge the gap and to enable a profile-based view composition, a consolation of research in the field of information needs is required to determine a mapping between the profiles and the available CI data to satisfy the individual information needs.

5 Conclusion

In this work, we revisited two approaches that can enable the integration, tailoring, and presentation of scattered data collected from modern CI environments. We shortly highlighted the potentials of the approaches and described the remaining challenges to bridge the gap between the information available and the information needed by a stakeholder.

References

[1] M. Brandtner, E. Giger, and H. Gall. Sqa-mashup: A mashup framework for continuous integration. *Information and Software Technology*, 2014.

[2] M. Brandtner, S.C. Müller, P. Leitner, and H.C. Gall. Sqa-profiles: Rule-based activity profiles for continuous integration environments. In *Proc. SANER*, pages 301–310, 2015.

[3] R.P.L. Buse and T. Zimmermann. Information needs for software development analytics. In *Proc. ICSE*, pages 987–996, 2012.

[4] M. Fowler. Continuous integration. 2006. `http://www.martinfowler.com/articles/continuousIntegration.html`.

[5] T. Fritz and G.C. Murphy. Using information fragments to answer the questions developers ask. In *Proc. ICSE*, pages 175–184, 2010.