Learning from Software Project Histories

Predictive Studies Based on Mining Software Repositories

Verena Honsel(✉), Steffen Herbold, and Jens Grabowski

Institute of Computer Science, University of Göttingen, Göttingen, Germany
{vhonsel,herbold,grabowski}@cs.uni-goettingen.de

Abstract. In software project planning project managers have to keep track of several things simultaneously including the estimation of the consequences of decisions about, e.g., the team constellation. The application of machine learning techniques to predict possible outcomes is a widespread research topic in software engineering. In this paper, we summarize our work in the field of learning from project history.

1 Introduction

The use of software repository data to investigate software evolution and for predictive studies is a wide-spread research topic, that spawned whole conferences like the MSR[1] in the software engineering community. The general approach depicted in Fig. 1 is similar for all application scenarios. First, the different data sources, i.e., repositories, are selected and the required data for the purpose of the investigation are combined. Researchers make use of the version control systems, issue tracking systems, mailing lists and similar systems as repositories. Second, a mental model of the software system is build which is filled with information from the repositories. These two steps can be summarized as data retrieval and modeling. Then, the usage of applicable tools for analysis accomplishes the mining process.

For almost a decade, our research group is interested in the application of theoretical methods to address problems from software repository mining.

- We applied a generalization of Probably Approximately Correct (PAC) learning to optimize metric sets [5].
- We worked on defect prediction in a cross-project context which leads to transfer learning problems [3,4].
- We developed an agent-based simulation model for software processes with automated parameter estimation [7,8].
- We created a model of the developer contribution behavior based on Hidden Markov Models (HMMs) [6].
- We implemented a smart data platform which can combine data collection and analysis through machine learning [9].

[1] http://msrconf.org/.

B. Berendt et al. (Eds.): ECML PKDD 2016, Part III, LNAI 9853, pp. 267–270, 2016.
DOI: 10.1007/978-3-319-46131-1_32

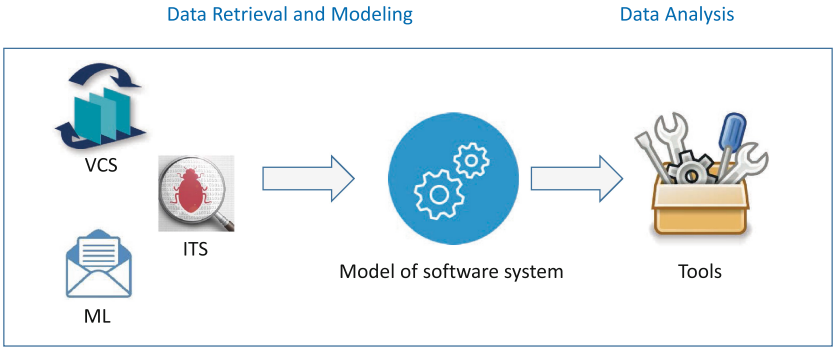

Fig. 1. Mining software repositories (adopted from D'Ambros et al. [2]).

2 Optimization of Metric Sets with Thresholds

Our first contribution is an approach to optimize metric sets for classification using a threshold-based approach. Threshold-based classifications are an important tool for software engineering as they are easy to interpret by both developers and project managers and can be used to, e.g., define coding guidelines. In our work [5], we demonstrate that very few metrics are sufficient to apply threshold based approaches, if the metrics are selected carefully and the thresholds are optimized for the smaller metric set. To achieve this, we use a combination of a brute force search of the potential metric sets combined with a generalized PAC learning approach [1] to determine optimal thresholds.

3 Cross-Project Defect Prediction

Accurate defect prediction can be used to focus the effort of quality assurance and, thereby, ultimately reduce the costs of a project while still ensuring a high quality product. Cross-project defect prediction deals with the problem of using data from outside of the project scope where the prediction is applied to, i.e., across project context. Hence, cross-project defect prediction is a transfer learning challenge. Within our work, we proposed an approach for improving prediction models based on selecting a subset of the training data through relevancy filtering [3]. Using the filtered training data, standard classification models, e.g., Support Vector Machine (SVM), Naïve Bayes, and Logistic Regression were used to predict defects. Moreover, we provided the research community with a tool to benchmark prediction results [4].

4 Software Process Simulation

For the simulation of software processes we consider several facets of software processes over time and their impact on software quality. The general idea for

building software process simulation models is to investigate repositories with the aim to find patterns which can describe evolutionary phenomena. For this, we applied statistical learning and machine learning, e.g., for the regression of growth trends. Our approach is agent-based, with the developers as active agents working on the software artifacts as passive agents. With our model, we simulated system growth, bugs lifespan, developer collaboration [7], and software dependencies [8].

5 Developer Contribution Behavior

Developers act in different roles in development projects, e.g., as core developer, maintainer, major developer or minor developer. We use of HMMs to describe involvement dynamics and the workload for the different developer types switching between different states (low, medium, and high) [6]. We take several actions of developers into account to model their workload: the monthly number of commits, bugfixes, bug comments, and mailing list posts. Figure 2 illustrates the learning process. We start with a sequence of monthly activity vectors as observations. We use the threshold learner described in Sect. 2 to classify the observations into low, medium, and high for each metric and with a majority vote for each observation. With the Baum-Welch and the Viterbi algorithm we calculate the transitions between the involvement states (e.g., low involvement to medium involvement) and the emissions for all states (i.e., the workloads). We build a HMMs for each developer of a project, as well as one general model for all developers. This way, we can describe the activity and workload of developers dynamically, which we will use to extend our simulation model to allow for changes in the project team during the simulation.

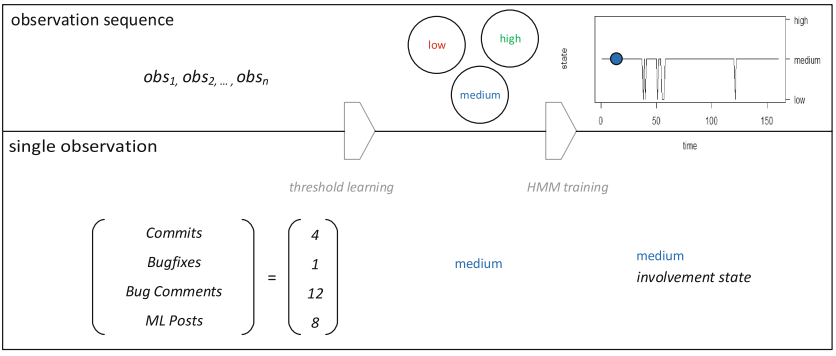

Fig. 2. Learning of developer's involvement state sequence.

6 Mining and Analysis Platform

A current problem in the state of practice of mining software repositories is the replicability and comparability of studies, which is a threat to the external

validity of results. To address this, we created the prototype SmartSHARK [9]. SmartSHARK mines data from repositories automatically and provides users with the ability to analyze the data with Apache Spark, a big data framework. Through the MLlib of Apache Spark, we enable users to perform machine learning tasks on the collected data. Current examples on how to use the platform are, e.g., different models for defect prediction as well as a simple approach for effort prediction. SmartSHARK is available as a scalable Cloud platform that will provide a constantly growing amount of project data and, thereby, enable large scale experiments. By sharing Apache Spark jobs, the research become replicable and comparable.

7 Conclusion

Within our research, we show the manifold possibilities to apply machine learning techniques to problems from software engineering, ranging from PAC learning to determine thresholds over the transfer learning challenge cross-project defect prediction to simulation parameter estimation and modeling developers through HMMs, culminating in a smart data platform for software mining. We invite machine learning researchers to use their expertise to advance the state of the art of software engineering.

References

1. Brodag, T., Herbold, S., Waack, S.: A generalized model of pac learning and its applicability. RAIRO - Theor. Inf. Appl. **48**(2), 209–245 (2014)
2. D'Ambros, M., Gall, H., Lanza, M., Pinzger, M.: Analysing software repositories to understand software evolution. In: Mens, T., Demeyer, S. (eds.) Software Evolution, pp. 37–67. Springer, Heidelberg (2008)
3. Herbold, S.: Training data selection for cross-project defect prediction. In: Proceedings of the 9th International Conference on Predictive Models in Software Engineering (PROMISE). ACM (2013)
4. Herbold, S.: Crosspare: a tool for benchmarking cross-project defect predictions. In: The 4th International Workshop on Software Mining (SoftMine) (2015)
5. Herbold, S., Grabowski, J., Waack, S.: Calculation and optimization of thresholds for sets of software metrics. Empirical Softw. Eng. **16**(6), 812–841 (2011)
6. Honsel, V.: Statistical learning and software mining for agent based simulation of software evolution. In: Doctoral Symposium at the 37th International Conference on Software Engineering (ICSE) (2015)
7. Honsel, V., Honsel, D., Grabowski, J.: Software process simulation based on mining software repositories. In: The Third International Workshop on Software Mining (2014)
8. Honsel, V., Honsel, D., Herbold, S., Grabowski, J., Waack, S.: Mining software dependency networks for agent-based simulation of software evolution. In: The Fourth International Workshop on Software Mining (2015)
9. Trautsch, F., Herbold, S., Makedonski, P., Grabowski, J.: Adressing problems with external validity of repository mining studies through a smart data platform. In: 13th International Conference on Mining Software Repositories (MSR) (2016)