Feedback in der Softwaretechnikausbildung: Automatische Analyse von studentischen Softwareprojekten für Lehrkräfte

Thema:
Feedback in der Softwaretechnikausbildung: Automatische Analyse von studentischen Softwareprojekten für Lehrkräfte
Art:
MA
BetreuerIn:
Alexander Bazo
BearbeiterIn:
Felix Riedl
ErstgutachterIn:
N.N.
ZweitgutachterIn:
N.N.
Status:
in Bearbeitung
Stichworte:
Software Engineering Education
angelegt:
2019-11-28
Antrittsvortrag:
2020-02-18
Abschlussvortrag:
2020-10-20

Hintergrund

Um die Qualität von Team- und Projektarbeiten in der Softwaretechnik-Ausbildung adäquat bewerten zu können reicht es oft nicht, ausschließlich das eingereichte Artefakt (z.B. Projektquellcode) zu begutachten. Neben der Qualität des finalen Artefakts spielen zum Beispiel auch durchgängige Prozessqualität, Koordination und Zeitmanagement eine wichtige Rolle (Kay et al., 2010). Zudem ist es notwendig Einzelleistungen zu identifizieren und deren Qualität hinreichend exakt einschätzen zu können (Jones, 2010).

Die mittlerweile weitverbreitete Verwendung von Systemen zur Versionskontrolle in studentischen Projekten kann dabei helfen, dieser Problematik beizukommen. Sie gewährt Lehrpersonen kontinuierlichen und unmittelbaren Einblick in Planungs- und Entwicklungsprozesse. Auch ermöglicht sie im Zweifel eine eindeutige Zuordnung von Contributions im Quellcode und eine frühzeitige Identifikation von Problemen während der Bearbeitung. Eine Einsicht der Rohdaten aus Versionskontrolle und PM-Tools kann hierfür bereits einen Überblick über die laufende Arbeit an einem Projekt bieten (Jones, 2010), ist jedoch auch aufwendig und bietet nur wenig detaillierte Einblicke.

Aus diesem Grund befassen sich zahlreiche Autoren mit der Frage wie sich die Artefakte studentischer Code Repositories zur Verbesserung der Lehre automatisiert auswerten und interpretieren lassen (u.A. Liu & Stroulia, 2003; Liu et al., 2004; Mittal & Sureka, 2014; Sprint & Conci, 2019). Die Schwerpunkte dieser Arbeiten sind breit gestreut und umfassen beispielsweise die Evaluation von Teamwork (u.A. Liu & Stroulia, 2003; Kay et al., 2010; Rastogi et al., 2013), individuellen Arbeitsgewohnheiten (u.A.Eyolfson et al., 2011; Kazerouni et al., 2017) oder typischen Fehlern beim Programmieren (u.A. Wittman et al., 2011). Die Autoren beschränken sich dabei allerdings nur auf einen speziellen Nutzungskontext, beispielsweise dem Einsatz in Kursen für Programmieranfänger (Kay et al., 2010) oder in Studienabschlussprojekten (Poncin et al., 2011).

Zielsetzung der Arbeit

Diese Masterarbeit beschäftigt sich daher mit der Frage wie ein generisches Analysetool für studentische Software-Repositories gestaltet werden muss, dass es sich zum Einsatz in unterschiedlichen Feldern des Softwaretechnik-Curriculums einsetzen lässt. Analysestrategien und –Algorithmen werden hierbei aus verwandten Arbeiten und einer Marktanalyse im Bereich VCS-Analysis Tools entnommen und mit Dozierenden im Feld der Softwaretechnik-Ausbildung auf deren Eignung evaluiert. Als zweiter Teil der Arbeit wird Konzept und Architektur des Analysetools erarbeitet. Besonderer Fokus liegt hierbei auf der modularen Erweiterbarkeit von Datenakquise, -auswertung und -visualisierung innerhalb des Systems. Abschließend wird die Eignung des Tools mit Dozierenden des Softwaretechnik-Curriculums evaluiert.

Konkrete Aufgaben

- Extraktion von Analysemethoden aus Related Work und bestehenden Tools
- Identifikation geeigneter Analysemethoden mit Lehrpersonal aus der Softwaretechnik-Ausbildung (Uni Regensburg, OTH Regensburg)
- Konzeption und Design des Anwendungsframework
- Implementierung des Frameworks
- Implementierung eines adäquaten Subsets an Analysemethoden
- Evaluation des Frameworks mit Lehrpersonal aus der Softwaretechnik-Ausbildung

Erwartete Vorkenntnisse

tba

Weiterführende Quellen

Kay, J., Koprinska, I., & Yacef, K. (2011). Educational data mining to support group work in software development projects (pp. 173-185). Boca Raton, FL: CRC Press.

Jones, C. (2010). Using subversion as an aid in evaluating individuals working on a group coding project. Journal of Computing Sciences in Colleges, 25(3), 18-23.

Liu, Y., Stroulia, E., Wong, K., & German, D. (2004, May). Using CVS Historical Information to Understand How Students Develop Software. In MSR (pp. 32-36).

Liu, Y., & Stroulia, E. (2003, November). Reverse Engineering the Process of Small Novice Software Teams. In WCRE (Vol. 3, p. 102).

Sprint, G., & Conci, J. (2019). Mining GitHub Classroom Commit Behavior in Elective and Introductory Computer Science Courses. The Journal of Computing Sciences in Colleges, 76.

Rastogi, A., Gupta, A., & Sureka, A. (2013, February). Samiksha: mining issue tracking system for contribution and performance assessment. In Proceedings of the 6th India Software Engineering Conference (pp. 13-22). ACM.

Kazerouni, A. M., Edwards, S. H., Hall, T. S., & Shaffer, C. A. (2017, June). DevEventTracker: Tracking Development Events to Assess Incremental Development and Procrastination. In Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education (pp. 104-109). ACM.

Eyolfson, J., Tan, L., & Lam, P. (2011, May). Do time of day and developer experience affect commit bugginess?. In Proceedings of the 8th Working Conference on Mining Software Repositories (pp. 153-162). ACM.

Wittmann, M. R. A., Bower, M., & Kavakli-Thorne, M. (2011, June). Using the SCORE software package to analyse novice computer graphics programming. In Proceedings of the 16th annual joint conference on Innovation and technology in computer science education (pp. 118-122). ACM.

Poncin, W., Serebrenik, A., & van den Brand, M. (2011, October). Mining student capstone projects with FRASR and ProM. In Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion (pp. 87-96). ACM.

Mittal, M., & Sureka, A. (2014, May). Process mining software repositories from student projects in an undergraduate software engineering course. In Companion Proceedings of the 36th International Conference on Software Engineering (pp. 344-353). ACM.