Daniel Schiffner (Hrsg.): Proceedings of DeLFI Workshops 2018
co-located with 16th e-Learning Conference of the German Computer Society (DeLFI 2018)
Frankfurt, Germany, September 10, 2018

Integration von DevOps in Lernplattformen

Wie die Programmierausbildung weiterentwickelt werden könnte und die didaktischen Implikationen

Paul Grimm[1]

Abstract: The further development in the working structures for software development should lead to integrate these structures into programming education programs: for example, by increasing the use of aspects of DevOps, an up-to-date approach for software development characterized through automatization and monitoring. Instead of creating artificial learning environments in which teachers give feedback to learners, novel learning environments could already base fully on processes and tools that play a major role in everyday work. A direct benefit could also be a better and faster feedback for the learners.

Abstract: Die Weiterentwicklung in den Arbeitsstrukturen für die Softwareentwicklung sollten dazu führen, dass diese Entwicklungen auch in die Programmierausbildung integriert werden, indem beispielsweise vermehrt Aspekte von DevOps, einem aktuellen Ansatz in der Softwareerstellung, der sich insbesondere durch Automatisierung und eines geeigneten Monitorings auszeichnet, in die Lehre und den dazu genutzten Lernumgebungen integriert werden. Statt mit Lernumgebungen nur künstliche Lerninseln zu schaffen, in denen Lehrende Feedback an Lernende geben, könnten neuartige Lernumgebungen bereits umfassender auf Prozessen und Tools basieren, die im beruflichen Alltag eine große Rolle spielen. Ein direkter Vorteil könnte auch ein besseres und schnelleres Feedback für die Lernenden sein.

Keywords: Programmieren, Programmierung, DevOps, Continuous Integration, Lernumgebungen, eLearning

1 Nutzung von DevOps für die Lehre

DevOps als aktueller Ansatz in der Softwareerstellung zeichnet sich insbesondere durch Automatisierung aus, die mit Hilfe einer entsprechenden Deployment-Pipeline umgesetzt wird und mit Hilfe eines geeigneten Monitorings überwacht wird [Ha15]. Die Deployment Pipeline basiert hierbei auf einem geeigneten Versionsmanagementsystem und integriert Schritte wie Code Review, Unit-Testing und Profiling ebenso um wie statische (wie z.B. Syntax Check) und dynamische Analysen (wie z.B. Lasttests oder Benchmarks).

[1] Hochschule Fulda, Angewandte Informatik, Leipziger Straße 123, 36037 Fulda, paul.grimm@cs.hs-fulda.de

Üblicherweise werden in der Programmierausbildung von den Lehrenden Aufgaben herausgegeben, die dann nach der Bearbeitung durch die Lernenden abgegeben bzw. vorgezeigt werden. Abschließend kann eine manuelle oder automatisierte Bewertung erfolgen. Hierzu können Lernplattformen wie Moodle [Mo18] in Kombination mit entsprechenden PlugIns wie CodeRunner [Co18] zum Einsatz kommen. Als Feedback erhält der Lernende somit entweder eine Antwort des Lehrenden oder die automatisierte Antwort. Konsequenterweise könnte ein Ansatz in der Programmierausbildung sein, dass anstelle bisherigen Abgabemöglichkeiten über Lernplattformen ein Versionsmanagement-system (wie z.B. Subversion [Su18] oder Git [Gi18]) zum Einsatz kommt, über das alle Abgaben zu erfolgen haben. Feedback könnte in diesem Sinne dank einem Automatisierungstools wie Jenkins [Je18] mit Hilfe von Unit-Tests und anderen Monitoringelementen erfolgen, die das toolgestützte manuelle Code Review unterstützen. Ähnlich wie bei PlugIns wie CodeRunner erhalten die Lernenden direkt bei der Abgabe durch die automatische Auswertung der Unit-Tests ohne Zeitverzögerung das Feedback über die Qualität ihrer Abgabe. Zudem lernen die Studierenden direkt die Vorteile einer testgetriebenen Entwicklung (Test-Driven-Developments, TDD) kennen.

Grundlegend muss betrachtet werden, welche Implikationen an die Didaktik durch die vorgeschlagene Vorgehensweise resultieren und welche Anforderungen daraus wieder an die technische Umsetzung abgeleitet werden können. Wenn für eine gesamte Gruppe ein gemeinsames Codeverwaltungssystem eingesetzt wird, so führt dies beispielsweise dazu, dass jeder Lernende Zugriff auf die Abgaben aller Anderen erhält. Der automatische Reflex hierbei ist, dass dies nicht möglich sein darf, um Plagiate bei den Abgaben zu verhindern. Andererseits wäre es wünschens- und erstrebenswert, dass Lernende die Lösungen der anderen mit der eigenen Lösung vergleichen. Nicht nur erhält hierdurch jeder Lernende gute Anregungen, er lernt auch, wie fremder Code betrachtet werden muss bzw. wie man sicherstellt, dass der eigene Code von anderen gelesen werden kann.

Interessant ist die Vorstellung, dass es für einen gesamten Studiengang oder einen Fachbereich ein einziges gemeinsames Versionsmanagementsystem gibt statt getrennte für einzelne Kurse. Hierüber würden Studierende nicht nur erlernen, wie große Repositories gut strukturiert werden können, es würden sich auch Synergieeffekte dadurch ergeben, dass Lernende wie Lehrende nicht jedes Mal eine neue Umgebung kennen lernen bzw. aufsetzen müssen und immer die gleiche zuverlässige Automatisierungs-Infrastruktur vorfinden würden. Dies würde soweit führen, dass diese auch für Projekte und Abschlussarbeiten der Lernenden genutzt werden könnten.

Das Ergebnis einer Integration von DevOps in Lernumgebungen hätte auf jeden Fall den Vorteil, dass Lernende von ersten Tag der Hochschulausbildung an langsam an die Arbeitsorganisation im späteren Berufsleben herangeführt werden können. Während zu Beginn das Feedback durch Code Review, Unit Tests und Monitoring die Lern-geschwindigkeit positiv beeinflussen könnten, kann im Verlauf der Programmierausbildung immer mehr erklärt werden, wie die eigentliche Lernumgebung aufgebaut ist und wie die Arbeitsprozesse durch den Aufbau unterstützt werden können.

Dieser Beitrag gibt Anregungen für die Weiterentwicklung von Lernumgebungen für die Programmierausbildung. Da die Programmierausbildung sehr vielfältig ist, konnten hierzu nur einzelne Ideen vorgeschlagen werden bzw. nur einzelne Aspekte betrachtet werden ohne dass daraus ein Anspruch auf Vollständig abgeleitet wird.

Literaturverzeichnis

[Ha15] Hasselbring, W.: DevOps: Softwarearchitektur an der Schnittstelle zwischen Entwicklung und Betrieb, 10.7.2015, http://eprints.uni-kiel.de/29215/1/2015-07-10Architekturen.pdf, Zugriff am 27.6.2018

[Mo18] Moodle, https://moodle.org, Zugriff am 27.6.2018

[Co18] CodeRunner PlugIn für Moodle, https://moodle.org/plugins/qtype_coderunner, Zugriff am 27.6.2018

[Su18] Subversion, https://subversion.apache.org, Zugriff am 27.6.2018

[Gi18] Git, https://git-scm.com, Zugriff am 27.6.2018

[Je18] Jenkins, https://jenkins.io, Zugriff am 27.6.2018