Deploy-to-Grading: Automatische Bewertung von Programmieraufgaben mit CI/CD-Pipelines

André Kirsch, André Matutat, Malte Reinsch, Birgit Christina George und Carsten Gips[1]

Abstract: Im Rahmen des Moduls Programmieren 2 des Studiengangs Informatik BA der Hochschule Bielefeld reichen Studierende seit mehreren Jahren ihre Programmierlösungen als Git-Pull-Requests ein und verlinken diese nur noch in ihrer Abgabe im Learning Management System. Bisher wurden die Lösungen der Studierenden anschließend in Präsenz mit den Lehrenden diskutiert und von diesen bewertet. Da diese Art der Bewertung viel Zeit in Anspruch nimmt, arbeiten wir an der Umstellung auf ein automatisches Bewertungssystem. Aus diesem Grund präsentieren wir in diesem Paper unser Konzept zur automatischen Bewertung von Programmieraufgaben mithilfe von Continuous Integration/Continuous Deployment-Pipelines. Im Gegensatz zu anderen automatischen Bewertungssystemen verwenden wir keine eigene Serverstruktur, sondern nutzen frei verfügbare Infrastruktur. Wir berücksichtigen dabei die einfache Übertragung auf andere Continuous Integration/Continuous Deployment-Pipelines sowie die Möglichkeit zur lokalen Ausführung.

Keywords: Autograding; Automatisiertes Feedback; CI/CD-Pipeline

1 Einleitung

Studierende erhalten in Programmiermodulen häufig umfangreiche praktische Aufgaben, um gelernte Inhalte anzuwenden. Um die Lösungen zu den Aufgaben zu bewerten, müssen Lehrende diese kontrollieren, was ein stark repetitiver Vorgang ist. Die manuelle Bewertung des Programmcodes erfordert häufig viel Zeit, was den Lehrenden die Möglichkeit nimmt, qualitative Probleme mit den Studierenden zu diskutieren oder individuelle Beratungen durchzuführen.

Im Rahmen des Moduls Programmieren 2 im Studiengang Informatik BA an der Hochschule Bielefeld möchten wir ein automatisches Bewertungssystem für Java entwickeln und einsetzen. Bei der Bearbeitung der Programmieraufgaben wird Git eingesetzt. Studierende bearbeiten die Aufgaben lokal und laden ihre Lösungen in ihr Team-Repository auf einem Git-Server (GitHub/GitLab/...). Dort erstellen sie zur Abgabe einen Pull Request (PR), welchen sie anschließend im Learning Management System (LMS) ILIAS verlinken.

Im Rahmen dieses Papers möchten wir ein Konzept vorstellen, durch das wir den bisherigen Abgabeprozess mit einer automatischen Bewertung verknüpfen. Studierende sollen weiterhin einen PR zur Abgabe erstellen. Commits zu diesem PR sollen automatisch eine Continuous Integration/Continuous Deployment (CI/CD)-Pipeline (hier konkret eine GitHub Action) ausführen, die den Bewertungsprozess durchführt. Studierende können in der Ausgabe der GitHub Action ihre erreichte Punktzahl einsehen. Ein Ziel des vorgestellten Konzeptes

[1] Hochschule Bielefeld, Campus Minden, Artilleriestraße 9, 32427 Minden, Deutschland, andre.kirsch@hsbi.de

doi:10.18420/abp2023-11

für ein automatisches Bewertungssystem soll die Zeitersparnis für Lehrende bei der Beurteilung von Programmierlösungen sein, wobei die Rückmeldungen bei der Begründung von Punktabzügen ausreichend ausführlich bleiben sollten. So bleibt für die Lehrenden mehr Zeit für die fachliche Diskussion mit Studierenden. Im Gegensatz zu anderen automatischen Bewertungssystemen nutzen wir keine eigene Serverstruktur und lehren gleichzeitig den Umgang mit Git und Free Open Source Software (FOSS). Daher ist unser automatisches Bewertungssystem eher für fortgeschrittenere Programmierkurse ab dem zweiten Semester geeignet. Die Implementierung des Konzepts soll in den nächsten Monaten stattfinden, sodass das System im Sommersemester 2024 erstmals eingesetzt und evaluiert werden kann.

Zusammenfassend ist der wesentliche Inhalt dieses Papers ein neues Konzept für die automatische Bewertung von Programmierlösungen unter Verwendung eines Versionskontrollsystems und einer CI/CD-Pipeline. Da die Implementierung des Konzeptes noch nicht abgeschlossen ist, wird nicht auf die konkrete Implementierung eingegangen und es können auch noch keine Erfahrungen zum Einsatz des automatischen Bewertungssystems präsentiert werden.

Im weiteren Verlauf des Papers werden in Kapitel 2 verwandte Arbeiten vorgestellt, wobei wir unser Konzept in das Themengebiet einordnen. Anschließend folgt in Kapitel 3 ein Überblick über unser aktuelles Abgabeverfahren. Dabei wird insbesondere auch darauf eingegangen, warum wir bei unserem automatischen Bewertungsverfahren einen Git-basierten Ansatz gewählt haben. In Kapitel 4 folgt eine detaillierte Erläuterung des erarbeiteten Konzepts. Das letzte Kapitel 5 schließt das Paper ab und gibt einen Ausblick auf die Umsetzung und den Einsatz des Deploy-to-Grading (D2G).

2 Verwandte Arbeiten

Automatische Bewertungssysteme werden schon seit langem international sowie in der deutschen Universitäts- und Hochschullandschaft genutzt, um mit wenig Zeitaufwand Programmierlösungen von Studierenden zu prüfen und zu bewerten. Beispiele sind ASB [HOS17] der Hochschule Trier, JACK [St17] von der Universität Duisburg-Essen und CodeOcean [St16] des Hasso-Plattner-Instituts, die vor einigen Jahren entwickelt wurden und stetig verbessert und erweitert werden. CodeOcean ist ein Autograder, mit dem Studierende durchgehend online arbeiten. Er stellt die Aufgaben zur Verfügung, die in einem Online-Editor gelöst werden. Das automatische Bewerten wird von derselben Weboberfläche aus ausgeführt. ASB und JACK kennzeichnen sich auch durch eine eigene Client-Server-Struktur. Die Lösungen werden von den Studierenden lokal erarbeitet und anschließend in die Webanwendung für die Bewertung hochgeladen. D2G soll im Gegensatz zu den genannten Autogradern keine eigene Client-Server-Struktur benötigen.

Neben den automatischen Bewertungssystemen hat der Git-Arbeitsablauf insbesondere in fortgeschrittenen Lehrmodulen Einzug gehalten, weswegen die Nutzung von Git in einem automatischen Bewertungssystem sinnvoll erscheint. GitGrade [Zh20] ist ein solches

Git-basiertes automatisches Bewertungssystem, welches an der University of Washington entwickelt wurde und auf einer universitätseigenen GitLab-Instanz aufsetzt. Es beinhaltet typische Bewertungsverfahren und unterstützt qualitatives Code Review-Feedback. Studierende können über eine eigene Weboberfläche Aufgabenstellungen akzeptieren, die sie in einem GitLab Repository bearbeiten und anschließend über dieselbe Weboberfläche einreichen können. Im Gegensatz zu GitGrade und anderen Autogradern soll D2G keine eigene Serverinstanz benötigen, sondern ausschließlich lokal oder als CI/CD-Pipeline ausgeführt werden. Dabei liegt ein Augenmerk auf der Portierbarkeit.

GitHub bietet mit GitHub Classroom [Gi23] selbst ein kostenfreies Tool zum Bewerten von Programmierlösungen an. Jede Aufgabe wird dabei in einem eigenen Repository verwaltet. Über ein Webinterface kann dieses Repository in eine Aufgabenstellung umgewandelt werden. Über einen Link können Studierende die Bearbeitung der Aufgabe starten. GitHub Classroom vereinfacht insbesondere die Verteilung der Aufgaben. Außerdem zeigt es automatisch eine Übersicht über die Lösungen an. Ein Nachteil von GitHub Classroom ist die fehlende Flexibilität bei der Bepunktung der Lösungen, da Teilpunkte bei Tests nicht vergeben werden können. Weitere Negativpunkte sind das Fehlen einer Repository übergreifenden Plagiatsprüfung sowie der Möglichkeit zur lokalen Ausführung. Des Weiteren versteckt es den Git-Workflow in großen Teilen. In unserem Konzept möchten wir die Flexibilität bei der Bepunktung beibehalten sowie eine Plagiatsprüfung durchführen können.

3 Bisheriges Vorgehen

Im Modul Programmieren 2 setzen die Studierenden aktiv Git und einen beliebigen Git-basierten Workflow für die Zusammenarbeit in Dreier-Teams ein. Dazu erstellen sie sich zuerst einen Fork des Vorgabe-Repositorys. Häufig ist dies für die Studierenden der erste Berührungspunkt mit Git und einer der verschiedenen Online-Plattformen wie GitHub oder GitLab. Die Studierenden arbeiten gemeinsam an einem Projekt und teilen ihren Arbeitsfortschritt innerhalb des Teams. Für die Abgabe erstellen sie einen öffentlich zugreifbaren PR innerhalb ihres Repositorys und geben die URL zu diesem PR im LMS als Link ab. Das Lösungskonzept und die Umsetzung (Code) werden im Praktikum den Lehrenden vorgestellt und von diesen bewertet.

Die Verwendung von Git hat mehrere Vorteile im Vergleich zur Abgabe von Lösungen im LMS. So erlernen Studierende tiefergehend den typischen Arbeitsablauf in Unternehmen und FOSS-Projekten. In PRs können die konkreten Änderungen eingesehen und direkt Feedback am Quellcode gegeben werden. Auch wird verhindert, dass unvollständige Projektordner abgegeben werden. Nach Kertész [Ke15] müssen sich Studierende initial erst an die Arbeitsweise gewöhnen, können dadurch aber nicht nur ihre Programmierfähigkeiten, sondern auch ihre Gruppenarbeitskompetenz verbessern. Möglich ist auch, dass Studierende aktiv an der Verbesserung des Lehrmaterials mithelfen, wenn dieses in einem Repository öffentlich verfügbar ist. Ein Nachteil dieses Verfahrens ist, dass für die Benotung ein Bezug zwischen einem GitHub-Account und einer reellen Person hergestellt werden muss. Aus

Gründen des Datenschutzes sollte dies nicht über GitHub geschehen, weswegen trotzdem ein LMS notwendig ist, über das in unserem Fall der PR verlinkt wird.

4 Deploy-to-Grading

Beim D2G soll der bisherige Arbeitsablauf von Studierenden zu großen Teilen beibehalten werden. Studierende arbeiten weiterhin in einem Fork des Repositorys, in dem die Aufgabenstellungen liegen. Sie bearbeiten diese Aufgaben in einer frei wählbaren IDE und laden ihre Änderungen über Git hoch. Zur Abgabe erstellen sie einen PR. Dabei und bei jedem weiteren Commit zu diesem PR wird D2G automatisiert gestartet. Dadurch, dass das D2G auch bei jedem weiteren Commit ausgeführt wird, haben Studierende bis zur Deadline die Möglichkeit, ihre Lösung zu überarbeiten und erneut prüfen zu lassen. Einen Überblick darüber verschafft Abbildung 1. Damit zielen wir auch darauf ab, den typischen Softwareentwicklungsprozess mit CI/CD im Unternehmens- und FOSS-Kontext abzubilden. Gleichzeitig soll unser System offen und modular sein, sodass auch andere Programmiersprachen und Bewertungsmetriken verwendet werden können.

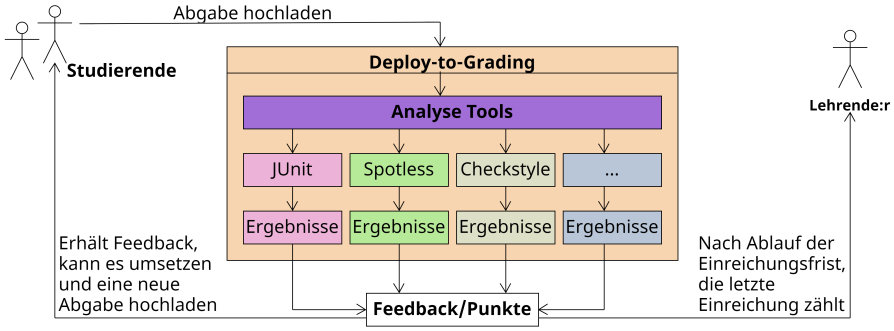

Abb. 1: Ablauf des Abgabeverfahrens von D2G. Studierende laden ihre Abgabe hoch, indem sie einen PR erstellen. Dadurch wird automatisch das D2G gestartet. Dieses führt für eine Aufgabe alle erforderlichen Tools aus und fasst dessen Ergebnisse zusammen. Daraus wird für die Studierenden ein Feedback und eine Punkteübersicht generiert. Studierende können bis zur Einreichungsfrist den Vorgang unbegrenzt wiederholen. Nach Ablauf der Einreichungsfrist kann der/die Lehrende die Ergebnisse einsammeln und für die Benotung nutzen.

Das D2G steht als GitHub Action in einem öffentlichen Repository zur Verfügung und wird über einen GitHub Workflow in das Studierenden-Repository eingebunden. Bei der Wahl der Onlineplattform und der eingesetzten Analysetools möchten wir flexibel sein und außerdem die lokale Ausführung ermöglichen. Durch dieses Konzept können Studierende und Lehrende auf eigene Ressourcen ausweichen (z. B. bei Datenschutzbedenken).

D2G checkt das zu prüfende Repository lokal aus und analysiert die relevanten Commits im Abgabeintervall. Anschließend wird das D2G für jede konfigurierte Aufgabenstellung

ausgeführt. Dabei werden zunächst alle Vorgabedateien mit den Dateien aus dem Vorgabe-Repository überschrieben, um ein Ändern dieser Dateien zu verhindern. Ausnahmen können hierbei in der Konfiguration angegeben werden. Anschließend folgt die Ausführung der Bewertungsmetriken. Bewertungsmetriken können dabei flexibel zu jeder Aufgabenstellung hinzugefügt und umfangreich konfiguriert werden. Die Ergebnisse werden in einem *results*-Ordner gesammelt, um sie zuletzt aufbereitet in der Konsole auszugeben und als Artefakt zur Verfügung zu stellen.

In den folgenden Unterkapiteln wird detaillierter auf einzelne Teile des D2G eingegangen. Dabei wird zuerst in Unterkapitel 4.1 ein Überblick über die Repository- und Dateistrukturen gegeben. Anschließend folgt in Unterkapitel 4.2 ein Überblick über die angewandten Bewertungsmetriken. Das Unterkapitel 4.3 beschreibt, wie die Ergebnisse des D2G den Studierenden und den Lehrenden übermittelt werden. Grenzen des Konzepts werden in Kapitel 4.4 aufgezeigt.

4.1 Aufbau der Repository- und Dateistrukturen

Da das D2G primär auf einem Git-Server genutzt werden soll, ist es naheliegend, auch die Aufgaben darüber zu verwalten. Wie in Abbildung 2 dargestellt, werden in einem nicht öffentlich sichtbaren Repository (*HomeworkSolution*) Aufgabenstellungen, Vorgabecode und Beispiellösungen gepflegt. Über *git subtree* können die Aufgabenstellungen und der Vorgabecode in ein für die Studierenden zugreifbares Repository *Template* exportiert werden. Es ist optional auch möglich, über *git filter-repo* einzelne Aufgaben in separate *Tasks*-Repositorys zu exportieren. Die Studierende erstellen vom *Template*- oder den *Tasks*-Repository(s) einen Fork, in dem sie die Aufgaben lösen und über den sie ihre Lösungen einreichen. Das *Template*- und alle darunter liegenden Repositorys sind mit unserem GitHub Workflow konfiguriert, welcher das D2G ausführt. Der Workflow verweist auf das *D2G*-Repository, welches das D2G als GitHub Action implementiert.

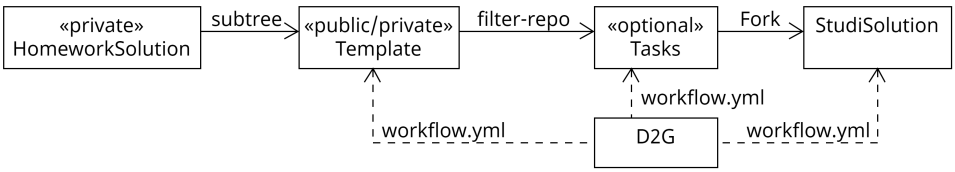

Abb. 2: Neben dem *D2G*-Repository für die D2G-GitHub-Action (CI/CD-Pipeline) existieren weitere Repositorys, die das D2G konfigurieren. Diese enthalten eine oder mehrere Aufgabenstellungen mit Vorgabecode. Im *HomeworkSolution*-Repository sind zusätzlich Beispiellösungen vorhanden. Das *StudiSolution*-Repository steht stellvertretend für alle Repositorys von Studierenden.

Für den Einsatz des D2G werden Konfigurationsdateien benötigt, die im selben Repository abgelegt werden wie die Aufgaben. Dazu gehört eine Konfigurationsdatei für das Aufgabenblatt sowie für jede Aufgabe eine eigene Konfigurationsdatei, in der die Bewertungsmetriken

konfiguriert werden. Die hier aufgezählten Metriken werden vom D2G ausgeführt. Alternativ prüfen wir die Unterstützung etablierter Konfigurationsformate, wie beispielsweise ProFormA 2.0 [Re19] und PEML [ME23].

4.2 Bewertungsmetriken

Bei der Auswahl an Bewertungsmetriken orientieren wir uns an typischen Bewertungsmaßstäben für Java-Quellcode und unterstützen unter anderem die statische sowie dynamische Codeanalyse. Dazu gehören beispielsweise die erfolgreiche Kompilierung, Unittests, CLI-Tests, Coding-Style, Dokumentation und Plagiatsprüfung. Des Weiteren planen wir die Umsetzung und Evaluation weiterer Blackbox-Testing-Möglichkeiten. Das D2G wird so umgesetzt, dass neue Bewertungsmetriken, insbesondere auch für andere Programmiersprachen, relativ einfach hinzugefügt werden können. Hervorzuheben ist, dass dieses Konzept mit entsprechenden Analysetools auch für Aufgabenstellungen jenseits der Informatik, z. B. für die Analyse von geschriebenen Texten, eingesetzt werden kann.

4.3 Rückmeldung an Studierende und Lehrende

Die Möglichkeiten zur Ausgabe der Ergebnisse an Studierende ist aufgrund der nicht genutzten eigenen Serverstruktur leicht eingeschränkt. D2G generiert eine Ausgabe der erreichten Punkte und einen Überblick über die Punktabzüge als Konsolenausgabe des GitHub Workflows. Da die Übersicht hier begrenzt ist, arbeiten wir außerdem an der Generierung eines Reports, der die Ergebnisse visuell aufbereitet. Zusätzlich werden die detaillierten Log-Files der Analysetools als Artefakt im GitHub Workflow den Studierenden zugänglich gemacht.

Lehrende können für das Erstellen einer Ergebnisübersicht D2G lokal ausführen und so die Ergebnisse der Studierenden datenschutzgerecht einsammeln und auswerten. Dazu wird der Anwendung eine aus dem LMS geladene Liste mit PR-URLs übergeben. Über die GitHub-API werden die Ergebnisse der PRs der Studierenden eingesammelt. Alternativ ist es auch möglich, den Bewertungsprozess erneut lokal auszuführen. Dies ist notwendig, wenn die Verfügbarkeit eines Artefakts abgelaufen ist. Zusätzlich kann eine Plagiatsprüfung durchgeführt werden. Abschließend wird eine *csv*-Datei mit den Punkten generiert, welche zurück in das LMS überführt werden muss.

4.4 Grenzen

Mit den Anforderungen der Verwendung öffentlich verfügbarer Infrastruktur und dem Git-basierten Workflow besitzt D2G Grenzen, die bei anderen automatischen Bewertungssystemen nicht zu finden sind. So ist man bei der Nutzung von D2G stark abhängig von der

öffentlich verfügbaren Infrastruktur, auf die man keinen Einfluss hat. Das System ist zwar so konzipiert, dass ein Wechsel zu anderen Anbietern leicht möglich sein soll, aber das kann mit zusätzlichem Arbeitsaufwand einhergehen. Außerdem besitzen GitHub, GitLab, usw. eine Begrenzung bei der Nutzung der Infrastruktur. Umgangen werden kann dieses Problem durch die Nutzung einer eigenen Server-Infrastruktur (z. B. eigene GitLab-Instanz).

Ein weiteres Problem sind die in der Regel öffentlich zugänglichen Studierenden-Repositorys, wodurch ein Kopieren der Lösungen von anderen Studierenden nicht ausgeschlossen werden kann. Zwar existieren Lösungen, um private Repositorys zu erstellen, dabei tauchen aber andere Probleme auf (z. B. erschwerter Zugriff durch die Lehrperson). Die eingesetzte Plagiatsprüfung sowie der öffentliche einsehbare Commit-Verlauf stellen zwar eine größere Hemmschwelle zum Kopieren von Lösungen dar, ob diese tatsächlich ausreichen, muss aber eine Evaluation zeigen. Möglicherweise ist auch eine Individualisierung der Aufgabenstellungen notwendig. Weiterhin ist aufgrund des Git-basierten Workflows ein Einsatz des D2G nur möglich, wenn im selben Modul oder in einem vorherigen Semester Git gelehrt wird, da das Abgabeverfahren ansonsten eine zu große Hürde für Studierende darstellen könnte.

Bei der Gestaltung von Übungsaufgaben ist im Vergleich zu Aufgabenstellungen ohne automatische Bewertung für die automatische Bewertung zusätzlicher Aufwand notwendig. Dazu gehören zum Beispiel die Erstellung von Unittests und die Bereitstellung von konkreten Vorgaben. Dabei ist darauf zu achten, dass die Vorgaben und Tests derart gestaltet werden, dass einerseits die automatische Testbarkeit gewährleistet ist, andererseits die Anzahl möglicher Lösungswege nicht zu sehr eingeschränkt wird.

5 Zusammenfassung und Ausblick

In diesem Paper haben wir mit D2G ein modulares und flexibles Konzept zur automatischen Bewertung von Programmierlösungen unter Gewährleistung des Datenschutzes vorgestellt. Es basiert auf dem Git-Workflow und nutzt CI/CD-Pipelines (GitHub Actions) zur Ausführung. Im Gegensatz zu anderen Lösungen verwenden wir keine eigene Serverstruktur, sondern nutzen frei zur Verfügung stehende Ressourcen. Des Weiteren bleiben wir mit der Umsetzung flexibel und ermöglichen somit die einfache Portierung (z. B. als Docker Container) auf unterschiedliche Git-Server sowie auch die lokale Ausführung. Gleichzeitig ermöglichen wir mit dem modularen Ansatz für Bewertungsmetriken die Erweiterung auf andere Programmiersprachen und Domänen.

Mit der Implementierung des vorgestellten Konzepts wurde bereits im Sommersemester 2023 begonnen. Wir planen, im darauf folgenden Wintersemester die Implementierung fertigzustellen und den praktischen Einsatz vorzubereiten (Überarbeitung der Aufgaben, Implementierung von Tests). Die neu entwickelten Aufgabenstellungen sollen sich spezifisch mit bestimmten Vorlesungsthemen auseinandersetzen. Zusätzlich sollen dann auch die Vorlesungsthemen in Sinne des Self-Paced-Learnings in einer relativ freien Reihenfolge bearbeitet werden können - das automatische Bewertungssystem soll diesen Spielraum

ermöglichen. Es soll im Sommersemester 2024 erstmalig produktiv eingesetzt und auch in der Lehrveranstaltung evaluiert werden. In der Evaluation möchten wir weitere Vor- und Nachteile des vorgestellten automatischen Bewertungssystems für die Lehrenden und insbesondere Studierenden ermitteln und prüfen, welchen Einfluss das System auf den Lernerfolg hat. Außerdem möchten wir umfangreiche Praxiserfahrung sammeln und das System weiter verfeinern. Der aktuelle Stand des Projektes befindet sich auf GitHub unter `https://github.com/Programmiermethoden/Deploy-to-Grading`.

Literaturverzeichnis

[Gi23] GitHub Inc.: GitHub Classroom, `https://classroom.github.com`, [Letzter Zugriff: 02.06.2023], 2023.

[HOS17] Herres, B.; Oechsle, R.; Schuster, D.: Der Grader ASB. In (Bott, O. J.; Fricke, P.; Priss, U.; Striewe, M., Hrsg.): Automatisierte Bewertung in der Programmierausbildung. Kap. 16, 2017, ISBN: 978-3-8309-3606-0.

[Ke15] Kertész, C.-Z.: Using GitHub in the classroom - a collaborative learning experience. In: IEEE 21st International Symposium for Design and Technology in Electronic Packaging (SIITME). S. 381–386, 2015.

[ME23] Mishra, D. S.; Edwards, S. H.: The Programming Exercise Markup Language: Towards Reducing the Effort Needed to Use Automated Grading Tools. In: Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1. S. 395–401, 2023.

[Re19] Reiser, P.; Borm, K.; Feldschnieders, D.; Garmann, R.; Ludwig, E.; Müller, O.; Priss, U.: ProFormA 2.0 – ein Austauschformat für automatisiert bewertete Programmieraufgaben und für deren Einreichungen und Feedback. In: 4. Workshop „Automatische Bewertung von Programmieraufgaben“. S. 43–50, 2019.

[St16] Staubitz, T.; Klement, H.; Teusner, R.; Renz, J.; Meinel, C.: CodeOcean - A versatile platform for practical programming excercises in online environments. In: IEEE Global Engineering Education Conference. S. 314–323, 2016.

[St17] Striewe, M.: Der Grader JACK. In (Bott, O. J.; Fricke, P.; Priss, U.; Striewe, M., Hrsg.): Automatisierte Bewertung in der Programmierausbildung. Kap. 9, 2017, ISBN: 978-3-8309-3606-0.

[Zh20] Zhang, J. K.; Lin, C. H.; Hovik, M.; Bricker, L. J.: GitGrade: A Scalable Platform Improving Grading Experiences. In: Proceedings of the 51st ACM Technical Symposium on Computer Science Education. 2020.