# Akzeptanz der Nutzung von automatisiertem Assessment im Rahmen einer virtuellen Vorlesung

Florian Horn[1], Daniel Schiffner[2] und Detlef Krömker[3]

**Abstract:** Durch die Umstellung auf virtuelle Lehre findet auch die Verwendung von automatischen Tools zur Bewertung von Programmieraufgaben immer mehr den Einzug in die Lehre. Im Rahmen einer solchen virtuellen Vorlesung wurde eine Bewertung durch die Studierenden vorgenommen, um daraus Erkenntnisse für die zukünftige Einbettung in der Lehre zu ziehen. Die Vorlesung zielt dabei auf höhere Semester des Bachelorstudiengangs ab und nutzt dabei Vorerfahrungen der Studierenden. Insgesamt wurde Feedback von 47 Studierende durch einen Fragebogen erhoben, und daraus Rückschlüsse auf die Qualität und Einsetzbarkeit von Unit-Tests gezogen.

**Keywords:** UEQ, Moodle, CodeRunner

## 1 Einleitung

Die Verwendung von automatisiertem Assessment ist nach wie vor mit einem hohen Aufwand für beide Seiten, Lehrende und Lernende, verbunden. Anstatt ein hochgradig spezialisiertes Werkzeug für ein bestimmtes Aufgabenformat zu nutzen, zielt die vorliegende Studie darauf ab, eine Antwort auf die Akzeptanz sehr einfacher Programmieraufgaben durch die Studierenden zu geben.

In dem Untersuchungsfall wurden, im Kontext einer Vorlesung über Computergrafik, Übungen mittels des Tools CodeRunner in Moodle umgesetzt. In den vorherigen (Präsenz-)Iterationen wurden vergleichbare Aufgaben gestellt, die jedoch zum Großteil manuell bewertet wurden, allerdings nicht als Programmieraufgaben, sondern als klassische Berechnungen durch die Studierenden. Hierbei wurde jedoch stets auf eine programmatische Lösung der Aufgaben hingearbeitet. Statt einer eigenständigen Prüfungsform, waren die Übungen begleitend und konnten als Bonus in einer Klausur bzw. mündlichen Prüfung angerechnet werden.

In der neuen Variante wird methodisch auf eine automatisierte Testung von Quellcode gesetzt, welche den Studierenden bereits aus vorhergehenden Semestern bekannt ist. Jedoch wird eine höhere Selbstständigkeit erwartet und die Aufgaben sind entsprechend auch aufwändiger gestaltet. Generell wurden dabei mehrere Lernziele definiert. Zum einen

[1] Goethe University, studiumdigitale, Robert-Mayer Str. 10, 60325 Frankfurt, horn@studiumdigitale.uni-frankfurt.de

[2] DIPF | Leibniz Institute for Research and Information in Education, IZB, Rostocker Straße 6, 60323 Frankfurt, schiffner@dipf.de

[3] Goethe University, studiumdigitale, Varrentrappstraße 40-42, 60486 Frankfurt, kroemker@studiumdigitale.uni-frankfurt.de

doi:10.18420/abp2021-4

sollen Studierende durch die Programmierarbeit die einzelnen Bestandteile der Rendering Pipeline besser verstehen, da sie von den abstrakten Konzepten eine reale Umsetzung erstellen. Zum anderen soll durch die Implementierungsarbeit eine längere Auseinandersetzung mit der Thematik bewirkt werden. Hierbei eignen sich Programmieraufgaben besonders gut, da eine visuelle Ausgabe grundlegende Eigenschaft der Computergrafik ist.

Im Rahmen einer Befragung werden die folgenden Hypothesen getestet und entsprechend die Fragen darauf ausgerichtet:

1. Die automatische Validierung wird auf Seiten der Studierenden als alternative Prüfungsform akzeptiert und kann entsprechend auch eingesetzt werden.
2. Studierende sind, durch die unmittelbare Rückmeldung, motivierter selbst aufwändige Aufgaben erfolgreich zu bearbeiten.

Zur Validierung unserer Hypothesen wurde eine repräsentative Umfrage (47 von 139, ca. 33 %) unter den Teilnehmern der Veranstaltung durchgeführt. Dazu wurde auf Basis des UEQ und einzelnen Vertiefungsfragen das Feedback zu der Verwendung des Tools im Rahmen des eingesetzten LMS eingeholt. Als Motivation wird den Studierenden ein Bonus in Höhe von 4 % der zum Bestehen notwendigen Punkte für die Bearbeitung des Fragebogens gegeben.

## 2 Verwandte Arbeiten

Moodle [Mo21] ist ein weit verbreitetes Open Source Learning Management System (LMS). Moodle ist mittlerweile eine Community Entwicklung und erlaubt es weltweit Lehrenden diverse Features und Plugins zu verwenden, um es ihren Kursen anzupassen. Die vorhandenen Fragetypen sind dabei oft auf geschlossene Formate eingeschränkt (SC, MC, Lückentexte). Durch Plugins (bspw. H5P), können andere Formate integriert werden.

CodeRunner [CR21, LH16] ist ein Plugin für das LMS Moodle und erlaubt es Lehrenden Programmieraufgaben in diversen Sprachen zu stellen und diese automatisiert testen zu lassen. Hierbei erlaubt CodeRunner sowohl einen vordefinierten Testablauf (welcher Unit-Tests durchläuft) zu verwenden, als auch einen eigenen zu schreiben, um so beispielsweise Monte-Carlo Simulationen auf studentischen Abgaben durchzuführen. Eine Beschränkung des vordefinierten Testablaufs ist es, dass lediglich Textvergleiche zwischen der erwarteten Ausgabe des Testfalls und der studentischen Abgabe möglich sind.

In [Ul18] geben die Autoren einen Überblick über Software zur automatischen Bewertung von Programmieraufgaben und deren Auswirkung auf Programmieranfänger. Die Autoren erreichen hierbei das Fazit, dass die automatische Bewertung von Programmieraufgaben gerade für Anfänger wichtig ist, da das sofortige Feedback bei der Fehlerbehandlung hilft und somit die Studierenden motiviert, Aufgaben in mehreren Versuchen vollständig zu lösen.

In [CE20] stellen die Autoren die Ersetzung ehemaliger Self-Assesments auf MC Basis durch automatisch bewertete Programmieraufgaben und den Effekt dieser Intervention auf die Studierenden vor. Verwendet wurde hierbei auch CodeRunner. In dieser Studie zeigt sich zwar ein negativer Effekt auf die Bewertung der Self-Assessment Aufgaben. Laut den Autoren ist dies der Fall, weil MC Aufgaben die Programmier-Skill überschätzen. Zusätzlich zeigt sich auch ein Anstieg der Noten in der finalen Klausur. Zudem Bewerteten die Studierenden das erhaltene Feedback als besser, da Sie feingranularer auf Ihre Fehler hingewiesen wurden.

Ein standardisiertes Evaluationswerkzeug zur Erhebung der Usability ist das User Experience Questionaire (UEQ) [UEQ21]. Das UEQ besteht aus 27 Likert Fragen, bei denen der User seine Meinung zwischen zwei Gegensätzen verortet, etwa „übersichtlich“ und „verwirrend“. Die so dargestellte Meinung wird Kreuzvalidiert und auf 6 User Experience Dimensionen abgebildet. Für unsere Studie verwendeten wir die Kurzversion [HST17], da Usability nicht im Fokus unserer Studie stand. Diese besteht aus lediglich 8 Fragen und bildet diese auf 2 Dimensionen ab: Pragmatische und Hedonistische Qualität.

## 3 Durchführung

Im Rahmen der Vorlesung „Einführung in die Computergraphik“ haben wir automatisierte Programmieraufgaben, als Ersatz für Übungsblätter, als Teil des Tutoriums eingesetzt. Thematisch behandelt diese Vorlesung vor allem algorithmische Grundlagen, Datenformate und Konventionen der Computergraphik. Ziel ist es hierbei Studierenden zu ermöglichen, auch komplexe Konzepte wie Ray-Tracing und Rendering Pipelines zu verstehen und selber umzusetzen.

Die Vorlesung lief vollständig Online ab. Hierzu wurde Moodle eingesetzt und Studierenden Vorlesungsaufzeichnungen, Folien, Zusatzmaterial, sowie die Übungen zur Verfügung gestellt. Unterstützend hatten Studierende die Möglichkeit an einer wöchentlichen Sprechstunde über die Übungsaufgaben teilzunehmen, sowie ein in Moodle integriertes Forum zu verwenden, in welchem sie sowohl von den Lehrenden, als auch von Kommilitonen Antworten und Feedback erhalten konnten. Weiterhin haben die Studierenden selbstständig einen Discord-Server aufgesetzt, um dort den Austausch abseits der Lernplattform zu koordinieren.

Als Programmiersprache wurde Python verwendet, da Studierende der Universität diese bereits in anderen Veranstaltungen eingesetzt haben. Zur Auslieferung der Fragen wurde das Moodle-Plugin CodeRunner verwendet.

Um einen Eindruck vom verwendeten Testsystem zu erhalten wurde den Studierenden zunächst eine Handreichung zur Verfügung gestellt. In dieser wurde ausführlich der Test und Abgabe Prozess in CodeRunner erklärt, sowie wie die Punktevergabe erfolgt. Zusätzlich hatten die Studierenden zwei einfache Aufgaben zur Verfügung, die sie jederzeit abgeben konnten, um sich mit dem Prozess vertraut zu machen. Generell wurde

CodeRunner nur zur Abgabe verwendet. Vom aktiven Programmieren innerhalb von Moodle wurde den Studierenden abgeraten, da grundlegende Hilfsmittel, wie Code-Completion, fehlen.

Aufgaben bestanden in der Regel aus mehrere Teilaufgaben, die gemeinsam ein funktionsfähiges Programm ergaben. Studierende hatten, je nach Komplexität, 1 bis 2 Wochen Zeit zur Bearbeitung. Bedingt hierdurch sahen wir von einem „all-or-nothing“ Grading ab und gaben anteilig Punkte auf bestandene Unit-Tests.

Zusätzlich war es Studierenden erlaubt mehrmals abzugeben, allerdings mit einem Punktabzug des erreichbaren Maximums. Studierende hatten zwei Abgaben ohne Punktabzug (eine Erst-Abgabe und einen Freiversuch). Jede weitere Abgabe zog 10 % vom erreichbaren Maximum ab. Hierbei wurde sichergestellt, dass die erreichte Maximalnote verwendet wurde.

Um die Meinung der Studierenden zum automatisierten Testen, in diesem Fall spezifisch mit CodeRunner, zu erheben, wurde eine Evaluationsstudie durchgeführt. Das Ziel dieser Studie war, die subjektive Meinung der Studierenden zum Einsatz von automatisierten Testsystemen in Tutorien und als Prüfungsersatz zu erhalten. Neben geschlossenen Fragen (siehe Tabelle 1) hatten die Studierenden die Möglichkeit, zu bestimmten Fragen offenes Feedback zu geben. Zusätzlich haben wir Fragen zur Erfassung der Usability des Systems eingesetzt, da eine schlechte Usability einen signifikanten Einfluss auf die Akzeptanz von Lern- und Prüfungssoftware hat.

Der Fragebogen wurde den Studierenden der Vorlesung angeboten und als zusätzliche extrinsische Motivation zur Teilnahme an der Befragung wurden den Studierenden Bonuspunkte (4 % der zum Bestehen notwendigen Punkte) gegeben. Es wurde hierbei darauf geachtet, dass die Umfrage vollständig anonymisiert und die Speicherung der Daten zwecks Bonuspunkte Vergabe geblendet ist.

## 4 Ergebnisse

An der Umfrage nahmen 49 Studierende teil. Insgesamt im Kurs eingeschrieben sind 139 Studierende, von denen circa. 74 aktiv, zur Zeit der Online-Befragung, am Übungsbetrieb teilgenommen haben. 2 der Umfragen wurden ausgenommen, da diese keine Frage beantwortet haben. Die Umfrage bestand aus mehreren Likert-Fragen, einigen Freitext fragen, sowie der Kurzversion des UEQ.

Die Likert-Fragen waren skaliert von 1: „Stimme gar nicht zu“ bis 4: „Stimme vollständig zu“. Zudem hatten Studierende die Möglichkeit eine Frage auch mit „Keine Angabe“ zu beantworten. Zusätzlich hatten Fragen 7-10 die Zusatzfrage: „Ich würde meine Meinung zu Frage X ändern, wenn...“ und Studierende hatten die Möglichkeit ihre Meinung zu elaborieren. Die Ergebnisse der Quantitativen Befragung findet sich in Tabelle 1 und Abbildung 1 für unsere Fragen und Abbildung 2 für die Ergebnisse des UEQ.

| **Frage** | **M** | **SD** | **N** | **E** |
|---|---|---|---|---|
| 1. Ich habe die Einführungen und Beispiele in CodeRunner durch die Veranstalter als hilfreich empfunden. | 3.38 | 0.74 | 45 | 2 |
| 2. Ich habe das direkte automatisierte Feedback von Coderunner als förderlich für den Lernerfolg erlebt. | 3.02 | 0.84 | 47 | 0 |
| 3. Auch negatives Feedback (Anzeige von Fehlern) habe es ich als hilfreich und motivierend empfunden. Ich habe meine Lösung dann meist soweit verbessert, bis jeder Testcase erfolgreich war. | 2.91 | 0.99 | 47 | 0 |
| 4. Der automatisierte Feedback verführt dazu, viel zu viel Zeit in die Bearbeitung der Aufgabe zu investieren, um auch die letzte Ungenauigkeit auszumerzen. | 3.06 | 0.91 | 47 | 0 |
| 5. In unserem Fall ist Coderunner so konfiguriert, dass es nach dem zweiten Versuch zum „Prüfen“ einen Punktabzug von den maximal erreichbaren Punkten gibt. Ich verstehe, warum dieser Abzug nötig ist. | 3.04 | 0.91 | 46 | 1 |
| 6. Ich finde den Punktabzug (Nr. 5) insofern akzeptabel. | 2.67 | 0.86 | 46 | 1 |
| 7. Ich lehne die Nutzung von CodeRunner (grundsätzlich) ab, weil es den Lernerfolg nicht steigert. | 1.72 | 0.95 | 46 | 1 |
| 8. Ich befürworte die Nutzung von CodeRunner zur persönlichen Information des Lernenden, also ohne Bonuspunkte zu vergeben oder es als Prüfungssystem einzusetzen. | 2.67 | 1.00 | 39 | 7 |
| 9. Ich befürworte die Nutzung von CodeRunner (einschließlich der benutzten Bewertungsmethode für die Ergebnisse), wenn damit Bonuspunkte in geringem Umfang vergeben werden. | 2.23 | 1.08 | 40 | 7 |
| 10. Ich befürworte die Nutzung von CodeRunner (einschließlich der benutzten Bewertungsmethode für die Ergebnisse) als (open book) Prüfungssystem, so wie in CG eingesetzt. | 3.31 | 1.01 | 45 | 2 |

Tab. 1: Ergebnisse der quantitativen Befragung. Angegeben sind der Mittelwert(M), die Standardabweichung(SD), die Antwortzahl(N) und die Anzahl der Enthaltungen(E). Die Skala bewegt sich von 1 (Ablehnung) bis 4 (Zustimmung).

Wie diesen Ergebnissen zu entnehmen ist, sind Studierende generell mit einer Portfolio Bewertung durch automatisierte Tests einverstanden. Besonders bezeichnend sind hierfür die Antworten auf Frage 1 und Frage 7.

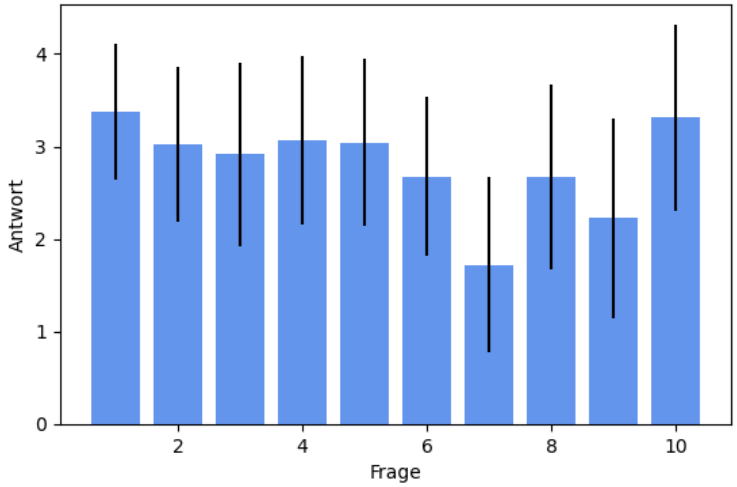


Abb.1: Ergebnisse der quantitativen Befragung. Angegeben ist für jede Frage der Mittelwert, sowie ihre Standardabweichung.

Der zweite Teil der Umfrage bezieht sich auf die einzelnen Erwartungshaltungen und Einsatzszenarien von automatisch getesteten Prüfungsaufgaben. Das Feedback der Studierenden zeigt, dass generell der Einsatz im Rahmen einer Vorlesung für fortgeschrittene Studierende hilfreich ist. Gleichfalls stimmen die meisten Studierenden dem gewählten Einsatz zu und finden dies als geeignete Prüfungsform (Fragen 2 und 7).

Für die Auswertung der UEQ Befragung (n=47) wurden die UEQ Auswertungs-Tools verwendet. Diese ergaben das CodeRunner leicht überdurchschnittlich abschneidet (vgl. Abb. 2). Entsprechend gehen wir davon aus, dass kein negativer Einfluss durch die Benutzung des CodeRunner entstanden ist. Dies ist vermutlich auch auf die dedizierte Verwendung (Nur Abgabe und Feedback) des Tools zurückzuführen.

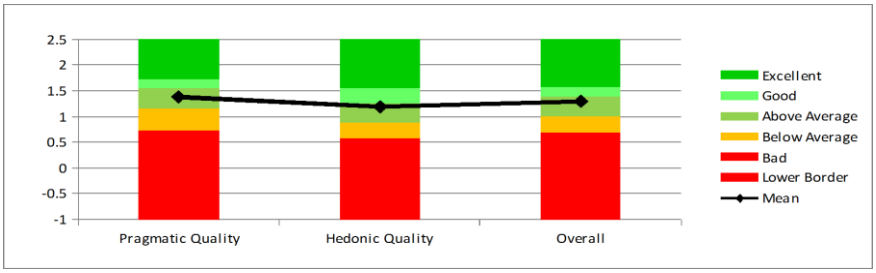


Abb. 2: Ergebnisse der UEQ Benchmark.

Die Umsetzung der Bewertung, und damit der ersten Hypothese, zeigt ein eher gemischtes Bild. Durch die Einschränkungen des Tools sind schnell Testfälle erzeugt, aber es werden lediglich textuelle Repräsentationen verwendet. Dies wird auch durch die Studierendenschaft moniert, welche sich teilweise detaillierte Tests wünschen. Dementsprechend zeigt sich auch ein eher unklares Bild in der Bewertung der Fragen 8, 9 und 10. Generell begrüßen die Studierenden, dass statt einer Klausur oder mündlichen Prüfung Programmieraufgaben, vergleichbar zu einem Praktikum, abzugeben. Sind aber unsicher, ob der Punkteabzug (Frage 6) so implementiert werden sollte. Die Motivation des Punkteabzugs widerrum wird unterstützt (Frage 5). Wir leiten aus den Fragen 8 und 9 entsprechend ab, dass eine automatisierte Bewertung für Übungsaufgaben nur dann sinnvoll ist, wenn auch ein klarer Einfluss in die Benotung vorhanden ist. Frage 10 bestätigt diesen Eindruck, ist aber auch mit einer generellen Prüfungsangst bei Studierenden konnotiert.

Zur Beantwortung der zweiten Hypothese zeigt sich eine positive Tendenz (Fragen 3 und 4). Die Erwartung, dass einzelne Studierende stets das Maximum der Punkte herausholen wollen zeigt sich auch in den offenen Antworten. Neben einzelnen, unzufriedenen Aussagen zeigt sich hier dennoch ein positiver Grundtenor, der sich insbesondere auch für ein individualisiertes Feedback ausspricht (Zitat: „Aufgaben mit weniger als 100 % Punk[t]zahl per Hand noch einmal evaluiert werden würden, um Leuten zu helfen, die eigentlich die richtige Idee hatten [...].“). Auch wäre eine geführte Herangehensweise teilweise erstrebenswert (Zitat: „Dann aber vielleicht begleitender, ähnlich eines Jupyter Notebooks. [...]“) Trotz der vielfältigen Angebote parallel zur Übung, scheinen einzelne Studierende ein weiteres Angebot zu vermissen (Zitat: „[...] Foren und Discord empfinde ich hierfür als sehr anonym und die Stimmung nicht immer wohlwollend. [...]“).

## 5 Fazit

Die Ergebnisse zeigen durchaus, dass Studierende der Informatik mit dem Konzept von automatisierter Bewertung von Aufgaben einverstanden sind. Es kristallisieren sich aber durchaus interessante Aspekte heraus, in denen sehr unterschiedliche Erwartungshaltungen der Teilnehmer als Ursprung zu vermuten sind.

Es zeigt sich ein unklares Meinungsbild, in dem die Teilnehmer die Art von Aufgaben als Prüfungs-Alternative zwar gerne annehmen und damit unsere erste Forschungshypothese unterstützen. Dennoch zeigt sich beim Einsatzszenario, bzw. den Möglichkeiten ein sehr unklares Bild. In vereinzelten Fällen werden hier starke Bedenken geäußert und mit einem Wunsch nach einer Vertiefung bzw. Flexibilisierung untermauert.

Für die zweite Forschungsthese hingegen zeigt sich eine positive Tendenz. Das unmittelbare Feedback hilft den Studierenden bei der Lösung von bisher nicht betrachteten Sonderfällen und schärft somit das Verständnis über die grundlegenden Algorithmen.

Ausgehend von den Erkenntnissen wird die Veranstaltungen in weiteren Iterationen angepasst, und vermehrt auch zu reinen Übungs- bzw. Self-Assessment-Zwecken automatische Aufgaben implementiert, welche dann auch eine tiefergehende Evaluation inkludieren sollen. Die Idee, personalisiertes Feedback auf die Abgaben zu geben könnte ihm Rahmen von beschränkten Aufgabenstellungen und bestimmten Programmierkonstrukten realisierbar sein. Weiterhin soll das Tool CodeRunner erweitert werden, um eine unmittelbare Interaktion zu ermöglichen und somit spezifisches Feedback und genaueres Testen zu ermöglichen.

## Literaturverzeichnis

[CE20] Croft, D.; England, M.: Computing with CodeRunner at Coventry University: Automated summative assessment of Python and C++ code. In Proceedings of the 4th Conference on Computing Education Practice 2020. 2020.

[CR21] CodeRunner, https://coderunner.org.nz/, Stand: 13.06.2021

[LH16] Lobb, R.; Harlow, J.: Coderunner: A tool for assessing computer programming skills. In ACM Inroads 7.1, S. 47-51, 2016.

[Mo21] Moodle, https://moodle.org/, Stand: 13.06.2021

[SHT17] Schrepp, M.; Hinderks, A.; Thomaschewski, J: Design and Evaluation of a Short Version of the User Experience Questionnaire (UEQ-S). In: IJIMAI 4 (6), pp. 103–108, 2017.

[UEQ21] User Experience Questionaire, https://www.ueq-online.org/, Stand: 13.06.2021

[Ul18] Ullah, Z. et al.: The effect of automatic assessment on novice programming: Strengths and limitations of existing systems. In Computer Applications in Engineering Education 26.6, S. 2328-2341, 2018.