Hans Pongratz, Reinhard Keil (Hrsg.): Die 13. E-Learning Fachtagung Informatik,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015 233

Bewertungsschema für eine abgestufte Bewertung von Programmieraufgaben in E-Klausuren

Nicole Jara[1] und Manuel Molina Madrid[2]

Abstract: Programmieraufgaben in E-Klausuren zu bewerten, ist bei der Entwicklung von automatischen Bewertungssystemen eine besondere Herausforderung. Es gibt unterschiedliche Lösungswege, in denen wiederum verschiedene Arten von Fehlern auftreten können. Ist es, unter diesen Umständen überhaupt möglich, Lösungen zu Programmieraufgaben fair und automatisch zu bewerten? In diesem Beitrag wird ein Bewertungsschema vorgestellt, welches gezielt für die automatische Bewertung von Programmieraufgaben in E-Klausuren konzipiert wurde. Das Bewertungsschema basiert auf sieben Typen von Syntaxfehlern, die in Abhängigkeit ihrer Schwere bei der Bewertung ein entsprechendes Gewicht haben. Die Grundidee dahinter ist, dass grundlegende Syntaxfehler bei der Bewertung höher gewichtet sind. Für die Fehlertypen wurden Programmieraufgaben alter Stift-und-Papier-Klausuren des Programmierkurses (Institut für Informatik, Universität zu Köln) untersucht und grundlegende Syntaxfehler bei Programmieranfängern identifiziert. Die Klassifizierung der Syntaxfehler ergab die Fehlertypen.

Keywords: Programmieraufgaben, Bewertungsschema, Bewertungssysteme, Syntaxfehlerklassen, E-Klausuren

1 Einleitung

In der Programmierausbildung an Hochschulen werden heute noch viele Klausuren mit Stift und Papier geschrieben. Bei der Korrektur des Lösungscodes werden Punkte in Abhängigkeit der Schwere des Fehlers von einer Gesamtpunktzahl abgezogen. In Veranstaltungen mit hohen Teilnehmendenzahlen gibt es Hilfskräfte, die nach einem vorgegebenen Schema die Arbeiten vorkorrigieren. Es kommt dennoch vor, dass für vergleichbare Fehler unterschiedliche Punkte vergeben werden. Eine einfache und wenig fehleranfällige Bewertung wäre: Programm läuft gleich volle Punktzahl, Programm läuft nicht keine Punkte. Wie ist das, wenn Studierende/r X nur eine Variabledeklaration als Lösungscode und Studierende/r Y ein Lösungscode abgibt, bei dem nur ein Semikolon fehlt? Eine abgestufte Bewertung des Lösungscodes würde der beschriebenen Situation Rechnung tragen.

Seit Bologna ist ein höheres Prüfungsaufkommen an Hochschulen zu verzeichnen. Klausuren bieten sich in Lehrveranstaltungen mir hohen Teilnehmendenzahlen als Prüfungs-

[1] Universität zu Köln, Programmierlabor des Instituts für Informatik, Pohligstr. 1, 50969 Köln, jara@informatik.uni-koeln.de
[2] Universität zu Köln, Programmierlabor des Instituts für Informatik, Pohligstr. 1, 50969 Köln, molina@informatik.uni-koeln.de

form an, da die Korrektur nach einem vergebenen Bewertungsschema erfolgen kann. Der Bewertungsaufwand bleibt weiterhin hoch. Zum Beispiel werden im Programmierkurs des Instituts für Informatik (Universität zu Köln) Klausuren von ca. 400 bis 600 Studierenden korrigiert. Es werden 8 - 12 Hilfskräfte für die Vorkorrektur eingesetzt, die gemeinsam mit dem Lehrenden bis zu vier 8-Stunden-Tage korrigieren. Die Klausuren sind sehr einfach gestaltet. Der Arbeitsaufwand ließe sich langfristig mit E-Klausuren verringern.

Wie ist der Stand von E-Klausuren in der Programmierausbildung? E-Klausuren in ILIAS [Ku05, ML14] oder E-Testate mit JACK [SG9] werden in der Programmierausbildung eingesetzt, wobei sie entweder auf das Schreiben von Programmcode oder auf eine abgestuften Bewertung verzichten. Für die abgestufte Bewertung ist ein Bewertungsschema erforderlich, das transparent, fair und eindeutig ist, um es in Bewertungssystemen einsetzen zu können. Ein solches Bewertungsschema zu entwerfen, basiert meist auf ein Identifizieren von Fehler, die von den Studierenden gemacht werden.

Die aktuellen Forschung basiert schwerpunktmäßig darauf, wie können Programmieranfänger beim Programmieren lernen unterstützt werden. Dabei werden bestehende Fehler identifiziert und den Studierenden Vermeidungsstrategien angeboten. Es geht dabei, um die Vermittlung von Lehrinhalten. Auf das Prüfen von Lehrinhalten in Klausuren wird nicht eingegangen. Dies würde sich auf den Datensatz auswirken. Es werden nicht einfach nur Programme von Studierenden während dem Lernprozess, sondern Programm aus einer Prüfungssituation (Klausur) betrachtet. Es gibt viele Forschungsberichte, die sich mit den auftretenden Programmierfehlern von Programmieranfängern befassen. Der Hintergrund dieser Analysen liegt in der Verbesserung der Lehre und nicht beim Prüfen. Es ist notwendig Programme von Studierenden zu untersuchen, die aus Klausurenlösungen stammen, um ein Bewertungsschema zu konzipieren.

In dieser Arbeit wird ein Bewertungsschema entworfen, das in einem Bewertungssystem eine abgestufte Bewertung von Programmieraufgaben ermöglicht. Zuerst wird der Korpus kurz vorgestellt und einige Hintergrundinformationen geben. Im Anschluss werden die Lösungscodes analysiert und Syntaxfehler identifiziert. Dann werden die identifizierten Syntaxfehler nach der Schwere klassifiziert, wobei grundlegende Syntaxfehler ein höheres Gewicht haben. Zum Schluss gibt es eine Zusammenfassung und einen Ausblick.

2 Datensatz

Der Datensatz umfasst Programmierkurs-Klausuren vom WS12/13 und WS13/14. Der Programmierkurs richtet sich an Studienanfänger und wird in der Regel im ersten Semester von den Studierenden belegt. Insgesamt wurden 74 Klausurlösungen der Studierenden digitalisiert, enthaltene Programmierfehler ermittelt und in Fehlerklassen statistisch festgehalten. Die Klausuren wurden dabei durch normalverteilte Zufallszahlen aus dem Klausurenpool ausgewählt. Die geprüften Klausuren lassen sich 48 männlichen

(64,86 %) und 26 weiblichen (25,14 %) Studierenden zuordnen. Der Anteil der Wirtschaftsmathematik Studierenden ist mit 47,30 % am höchsten, gefolgt von 40,54 % Wirtschaftsinformatikern, 10,81 % Mathematikern und 1,35% Physikern.

Die untersuchten Klausuraufgaben beinhalten das Programmieren einer Rekursion und Iteration, einer Matrizenaddition und das Umwandeln von Zahlensystemen mittels vorgegebener Methoden. Des Weiteren wird die Objektorientierung in einer Aufgabe behandelt, indem eine Klasse implementiert werden muss und anschließend ein Objekt der Klasse erzeugt und in einer LinkedList gespeichert werden soll.

3 Auswertung - Programmierfehler

Bei der Ermittlung von Syntaxfehlern in den Klausurenlösungen der Studierenden können ähnliche Fehlertypen wie bei bisherigen Forschungsansätzen, beispielsweise [AB15], [Br14] und [Hr03], ermittelt werden. Die zehn häufigsten Fehler sind in Tab. 1 aufgelistet.

Fehlertypen	% aller Fehler
Geschweifte Klammer nicht geschlossen	45,95
Variable wird nicht deklariert	20,27
Fehlendes `return` Statement	17,57
Parameterübergabe von `String` ohne ”…”	12,16
Kein generischer Datentyp angegeben	12,16
Runde Klammer nicht geschlossen	12,16
Länge des Arrays mittels `array.length()`	10,81
Deklarierung der Laufvariable fehlt	10,81
Methodenaufruf ohne Parameterklammern	9,46
Variablennamen werden doppelt	8,11

Tab. 1: Die zehn häufigsten Fehler

Aufgrund der geringen Aufgabenvielfalt und dem Prüfungskontext können spezifische Fehler festgehalten werden, wie der Aufruf von `array.length()`. Die Programme der Studierenden wurden in einer Prüfungssituation verfasst. Dies äußert sich auch in der Art und Weise, wie die Studierenden vorbereitet und die Programme aufgebaut sind. Bei dem Auswerten der Klausurenlösungen der Studierenden kann als Beobachtung festgehalten werden, dass die Lösungsformen der Programmieraufgaben keine hohe Variation aufweisen. Viele Studierende haben die gleiche Implementierung angegeben. Die Programme stimmen in den Bezeichner der Variablen, der Berechnung und dem Aufbau exakt überein. Diese Tatsache liefert den Verdacht, dass viele Studierende Lösungen aus ähnlichen Übungen oder alten Klausuraufgaben auswendig lernen und in der Klausur niederschreiben. Dieser Verdacht wird dadurch verstärkt, dass existierende Übungslösungen nicht der Klausuraufgabenstellung angepasst werden, sondern mitunter von den Studierenden unverändert hingeschrieben werden. Dadurch entstehen inhaltliche Fehler,

aber auch syntaktische Fehler treten gehäuft auf. Durch ein Auswendiglernen von kompletten Programmabläufen werden die Inhalte nicht verinnerlicht. Die Bedeutung einzelner Bestandteile ist für den Studierenden nicht bekannt, sodass beispielsweise keine Initialisierung einer Variablen durchgeführt wird.

4 Bewertungsschema

Um Fehler von Studierenden fair bewerten zu können, muss die Ursache für das Auftreten eines Fehlers in das Bewertungsschema miteinbezogen werden. Ziel der Bewertung einer Klausur ist das richtige Einstufen der Programmierkenntnisse eines Studierenden. Daher ist es notwendig, beispielsweise einen Flüchtigkeitsfehler wie das Vergessen eines Semikolons nicht zu hoch einzustufen und zu stark zu bestrafen. Dieser Fehler tritt häufig auf und würde daher schnell eine negative Auswirkung auf die Gesamtbewertung eines Studierenden haben. Solche Fehler sind nicht zwingend aussagekräftig im Zusammenhang mit den Programmierkenntnissen und würden zu einer verfälschten Bewertung der Programmierfähigkeiten führen. Aufgrund dieser Argumentation wird die unterste, am niedrigsten bewertete, Fehlerklasse durch sogenannte Flüchtigkeitsfehler gebildet, welche nicht als fundamental für das Verständnis der Programmierung gelten.

Die weitere Einteilung und Bildung von Fehlerklassen in dem hier vorgestellten Bewertungsmodell basiert auf dem Grundgedanken, dass fundamentale, d. h. grundlegende, Fehler der Programmierung höher bestraft werden. Viele weiterführende Kenntnisse in der Programmierung bauen auf Grundkenntnisse und einem Grundverständnis des Programmflusses auf. Daher ist es notwendig, ein fundiertes Wissen über den grundlegenden Aufbau und Datenfluss eines Programmes zu besitzen und einfache Datenstrukturen sicher verwenden zu können. Die Einstufung von Fehlern nach diesem Prinzip soll den grundlagenbasierten Schwerpunkt für das Erlernen einer Programmiersprache verdeutlichen und bei der Ermittlung und Bewertung der Programmierkenntnisse helfen. Um eine Programmiersprache zu beherrschen, ist es notwendig die Grundlagen verinnerlicht zu haben, d. h. die Syntax und den Einsatz von konkreten Programmbausteinen und den Datenfluss zu verstehen und anwenden zu können. Beispielsweise werden Fehler im Aufbau einer `for`-Schleife höher bewertet als Fehler zurückführend auf die Klassenvererbung.

Abb. 1 zeigt die Einteilung der Fehler in sieben Fehlerklassen. Die oberste Klasse „Grundlegendes Verständnis über Datenfluss und Aufbau“ stellt dabei die höchste und damit schwerwiegendste Fehlerklasse dar. Die Einteilung verläuft von der höchsten zur niedrigsten Fehlerklasse „Flüchtigkeitsfehler“. In jeder Klasse sind kursiv gedruckte Fehler abgebildet. Dabei handelt es sich um Fehler, die in der hier untersuchten Auswertung der Klausurlösungen aufgetreten sind. Sie wurden in das Konzept der Fehlerklassen eingeordnet und können als Beispiele zur Veranschaulichung dienen.

Grundlegendes Verständnis über Datenfluss und Aufbau Datenstrukturen deklarieren, initialisieren Programmstruktur	**Deklarierung und Initialisierung von Variablen** *Fehlende Deklarierung vor der Initialisierung* *Verwenden einer nicht initialisierten (nicht deklarierten) Variablen* *Gleicher Variablenbezeichner für mehrere unterschiedliche Variablen* **Wertzuweisung** *Nicht Zuweisungskompatibel* **Grundstruktur von Java-Programmen** Klassenkopf und main-Methode Klassenkopf und main-Methode *class fehlt*
Strukturelle Erweiterung eines Programmablaufs Entscheidungszweige Schleifen	**Verzweigung** *Aufbau einer if-else Anweisung* **Boolische Ausdrücke** *einfaches Gleichheitszeichen* **Schleife** Abweißschleife while, Durchlaufschleife do-while, Zählschleife for Schleifenkopf fehlerhaft *Initialisierungsteil: Deklarierung der Variable fehlt* Abbruchbedingung, Inkrementierungsteil
Zusammenfassen bestimmter Befehlsfolgen in Methoden Struktur und Aufbau einer Methode Aufruf einer Methode	**Methoden** Methodenkopf *Parameterklammern fehlen* *Parametertyp der Parameter fehlt* *Rückgabetyp fehlt* *Geschweifte Methodenklammern werden nicht gesetzt* Speziell Funktion (Methode mit Rückgabe) *return-Anweisung fehlt* *Datentyp der return-Anweisung stimmt nicht mit gefordertem Rückgabetyp überein* Platzierung: return nur in if-Bed., (return in Schleifen -semantischer Fehler) Methodenaufruf *Parameterklammern fehlen* *Parameterübergabe fehlt* *Parameterübergabe mit Angabe des Datentyps* *Methodeaufruf ohne Zuweisung des Rückgabewertes*
Array als spezieller Datentyp	**Arrays** *Array nur deklariert* *Länge des Arrays als Methodenaufruf a.length()* Fehlerhafter Zugriff auf Inhalte *Bezeichner hinter eckigen Klammern* *Operationen auf ganze Arrays angewandt* Mehrdimensionales Array: Deklarierung, Erzeugung , Zugriff analog zu eindim.
Objektorientierung	**Klassen** Konstruktor Erzeugung von Objekten *Generelle Struktur nicht einbehalten* *Argumente werden nicht übergeben* Objektorientierung *Methodenzugriff ohne Objektbezeichner* (weitere Fehler hinsichtlich Methode, s.o)
Erweiterung des Grundwissens	**Dynamische Datenstrukturen** LinkedList<T> *kein generischer Datentyp angegeben* *Erzeugung der Datenstruktur fehlerhaft* *Hinzufügen eines Elements nicht korrekt* **Vererbung** *Geerbte Klasse versucht auf private Attribut der Oberklasse zuzugreifen* **Abstrakte Klassen**
Flüchtigkeitsfehler	**Klammersetzung** *Runde Klammer ..) wird nicht geschlossen* *Geschweifte Klammer ..} wird nicht geschlossen* **Semikolon** Semikolon am Ende einer Anweisung fehlt

Abb. 1: Abgestufte Fehlerklassen zur Bewertung von Programmierfehlern

Wie schon angeführt, handelt es sich bei der obersten Klasse um die schwerwiegendsten Fehler: der Umgang mit Variablen und die Grundstruktur eines Programms. Der Studierende sollte gelernt haben, dass bevor eine Variable verwendet werden kann, muss diese deklariert und initialisiert worden sein. Des Weiteren sollte dem Studierenden bekannt sein, dass ein Java-Programm in eine Klasse geschrieben wird und eine main-Methode notwendig ist, um ein Programm auszuführen. Eine strukturelle Erweiterung des grundlegenden Programms stellt die Verzweigung und Schleife dar. Dazugehörend ist das Vergleichen mit Hilfe eines booleschen Ausdrucks. Aufbauend auf den bisherigen Elementen können mittels Methoden Programmabschnitte zusammengefasst werden. Ohne das Verständnis von Methoden ist keine Basis für die Objektorientierung geschaffen. Aus diesem Grund sind die Fehler eindeutig höher eingestuft als beispielsweise Fehler beim Erzeugen eines Objektes.

Arrays werden als spezieller Datentyp angeführt und erweitern die bisherigen Kenntnisse über den Umgang mit Variablen. Als nächste Fehlerklasse folgt die Objektorientierung, d. h. der Aufbau und Umgang mit Objekten. Die letzte reguläre Klasse beinhaltet sogenannte Elemente, die als Erweiterung des Grundwissens zusammengefasst werden und zum Ende des Programmierkurses vermittelt werden. Die LinkedList als dynamische Datenstruktur, die Vererbung und abstrakte Klassen bilden die genannte Fehlerklasse. Als abschließende Klasse dient die oben erwähnte Klasse der Flüchtigkeitsfehler. Darunter fällt z.B. das Vergessen eines Semikolons an Anweisungsende oder das Vergessen des Schließens von geschweiften und runden Klammern.

Anhand dieser Einteilung von Fehlern soll das grundlegende Prinzip der höheren Bestrafung von Grundlagenfehlern deutlich werden. Je tiefer die Fehlerklasse in dieser Abstufung abgebildet ist, desto weniger elementar ist sie für einen einfachen Programmablauf. Es lässt sich jedoch festhalten, dass sie inhaltlich trotzdem bedeutend für die Lösung der Klausuraufgaben ist. Fehler in niedrigeren Klassen beziehen sich auf Zusatzwissen oder weiterführendes Wissen, welche möglicherweise weniger oft angewandt wurden und in dessen Bereich ein Fehler weniger schwerwiegend eingestuft wird.

Durch die hier durchgeführte Abstufung von Fehlern wird der Kern der Java Programmierung in das Zentrum der Bewertung gestellt. Wie in Kapitel 3 aufgeführt, lassen sich einige Fehler auf ein Auswendiglernen von kompletten Programmabschnitten der Studierenden zurückführen. Durch die hier dargelegte Klassifizierung werden einige daraus folgende Fehler höher bestraft, wie beispielsweise die häufige Verwendung der nicht initialisierten Variablen `n`.

5 Zusammenfassung und Ausblick

Das in dieser Arbeit vorgestellte Bewertungsschema wurde auf Grundlage einer Analyse von Studierendenlösungen von Programmieraufgaben alter Klausuren des Programmierkurses am Institut für Informatik (Universität zu Köln) entworfen. Bei der verwendeten Programmiersprache handelt es sich um Java. Das entwickelte Bewertungsschema ba-

siert auf sieben Fehlertypen. Jeder Fehlertyp umfasst konkrete Syntaxfehler, welchen eine ähnliche Schwere bei der Bewertung zugeordnet werden kann. Grundlegende Fehler werden dabei höher gewichtet als weiterführende Fehler. Die Gewichtung der Fehlertypen kann durch entsprechende Klausurpunkte umgesetzt werden. Durch die abgestufte Klassifizierung ist das Bewertungsschema transparent und fair.

Bei der in diesem Semester durchgeführten zweiten Programmierkursklausur wird dieses 7-Fehlertypen-Schema an einer Stichprobe erstmalig eingesetzt und getestet. Das Schema ließe sich für weitere Arbeiten auch auf andere prozedurale und objektorientierte Programmiersprachen anpassen. Für die automatische, abgestufte Bewertung durch ein Bewertungssystem wird für jeden Fehler eine Bibliothek mit Syntaxfehlern angelegt, mit denen der durch das Bewertungssystem identifizierte Fehler übereinstimmen muss. Eine Herausforderung bleibt die automatische Identifizierung des Syntaxfehlers. Dieses Problem ist Gegenstand einer anderen Arbeit.

Literaturverzeichnis

[AB15] Altadmri, A.; Brown, N.C.C.: 37 Million Compilations: Investigating Novice Programming Mistakes in Large-Scale Student Data. In Proceedings of the 46th ACM Technical Symposium on Computer Science Education. ACM, Kansas City, 522-527, 2015.

[Br14] Brown, N.C.C. et.al.: A Large Scale Repository of Novice Programmers' Activity. In Proceedings of the 45th ACM Technical Symposium on Computer Science Education. ACM, Atlanta, 223-228, 2014.

[Hr03] Hristova, M. et.al.: Identifying and Correcting Java Programming Errors for Introductory Computer Science Students. In Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education. ACM, Reno, 153-156, 2003.

[Ku05] Küstermann, R. et.al.: Durchführung einer Online-Klausur mit ILIAS. Vortrag bei der 4th International ILIAS Conference, Nürnberg, 2005.

[ML14] Molina Madrid, M.; Lohmann M.: E-Klausuren mit Programmieraufgaben und automatischer Bewertung des Programmcodes im E-Learning-System ILIAS. Vortrag beim e-Prüfungs-Symposium, Aachen, 2014.

[SG09] Striewe, M.; Goedicke, M: Effekte automatischer Bewertungen für Programmieraufgaben in Übungs- und Prufungssituationen. In (Schwill, A. und Apostolopoulos, N. Hrsg,): Proceedings der 7. E-Learning Fachtagung Informatik, Lecture Note 153, 2009.