Misfit-Lehrevaluation: Perspektivwechsel und neue Lehrverbesserungsideen

Thomas Kirchmeier

Zusammenfassung: *Die Lehrevaluation ist einer der wichtigsten Pfeiler zur Verbesserung der eigenen Lehre. Durch die Anonymität der Teilnehmenden wird freie Meinungsäußerung ermöglicht. Doch wenn nur ein kleiner Teil der Studierenden an der Lehrevaluation teilnimmt und die Rückmeldungen fast ausschließlich positiv sind, wird die Wahrnehmung der Dozierenden verzerrt. Zum einen besteht dann scheinbar keine Notwendigkeit für Verbesserungen und zum anderen lassen sich aus überwiegend positiven Rückmeldungen kaum konkrete Verbesserungsvorschläge ziehen. Um die Lehrveranstaltung im Rahmen einer Evaluation dennoch kritisch zu hinterfragen, können die Misfits von EMPAMOS weiterhelfen. Der vorliegende Artikel zeigt eine Möglichkeit, wie diese Misfits in der Lehrevaluation angewendet werden können. Dabei wird auf eine ausgewogene Betrachtung aller Misfits geachtet und es werden auch nur die statistisch relevanten Misfits identifiziert.*

Abstract: *Evaluation is an important tool to improve one's own teaching methods. The participants' anonymity allows for free expression of opinions. However, if only a small number of students take part in an evaluation and their feedback proves to be almost exclusively positive, this may distort teachers' perception. On the one hand, there seems to be no need for improvement, while on the other hand, it is difficult to draw helpful suggestions for improvement from predominantly positive feedback. In order to still critically evaluate the course, the EMPAMOS misfits can prove useful. This article presents an approach to using the EMPAMOS misfits in teaching evaluations. In doing so, attention is paid to a balanced consideration of all misfits, and only the statistically relevant ones are identified.*

Schlagworte: *Lehrevaluation, Programmiersprachen, Misfits in der Evaluation, Game Thinking*

1. Die Ausgangssituation: Herausforderungen im Unterrichten von Programmiersprachen

Programmiersprachen wie Python oder Java sind in MINT-Studiengängen fester Bestandteil des Grundstudiums. Kenntnisse in Programmierung sind nicht nur für den späteren Beruf von entscheidender Bedeutung, sondern dienen auch während des weiteren Studienverlaufs als wertvolles Werkzeug zur Auswertung und Darstellung von Daten. Das Erlernen der Programmieransätze und vor allem des abstrahierenden Vorgehens bei der Aufgabenbearbeitung ist für Neulinge jedoch eine Herausforderung (Lahtinen et al., 2005; Dale & Weems, 2014; Konecki, 2014; Cheah, 2020; Kadar et al., 2021; Xue et al., 2024).

Cheah (2020) führt die Lernherausforderung für Studierende auf vier Faktoren zurück: die Programmierung selbst, die Problemlösungsfähigkeit, ineffektive Pädagogik sowie die persönlichen Eigenschaften und Einstellungen der Studierenden. Die ersten beiden Faktoren bedingen sich gegenseitig im vorangeschrittenen Unterricht (Dale & Weems, 2014), denn erst ein tieferes Verständnis des Implementierungsdesigns einer Aufgabe ermöglicht den Einsatz bestimmter Methoden wie Schleifen, Bedingungen oder gar Objekten (im Sinne der objektorientierten Programmierung). Diese Methoden sind zu Beginn oftmals verwirrend für die Studierenden, da es ihnen (noch) an Vorstellungskraft für die Ausführung der Programmierbefehle mangelt. Iteratives Ausprobieren und Erleben der Programmiersprache über verschiedene Aufgaben ermöglicht es ihnen, diese Vorstellungskraft nach und nach zu entwickeln (Edwards, 2003).

Der dritte Faktor, die ineffektive Pädagogik, ist auf überholte Lehrmaterialen zurückzuführen (Savage & Piwek, 2019), darunter Bücher und PowerPoint-Präsentationen. Der daraus abgeleitete Lehrplan nach Soloway und Spohrer (2013) betrachtet in den meisten Fällen nur das Basiswissen der verschiedenen Programmierkonzepte weitestgehend isoliert anhand von Ablauf-, Struktur- und Datenflussdiagrammen. Der Fokus liegt dabei auf Syntax und Semantik, jedoch nicht auf der Lösung komplexerer Probleme. Durch eine fehlende dynamische Interaktion der Studierenden mit der Programmiersprache wird zudem das spielerische Ausprobieren verhindert, was insbesondere beim Verstehen objektorientierter Programmiersprachen zu großen Hürden führt.

Bezogen auf die persönlichen Eigenschaften und Einstellungen stehen die Studierenden wiederum meist vor der Herausforderung, dass die abstrakte Programmierung nicht an früher erworbenes Wissen anknüpft (Cheah, 2020). Durch Austausch mit älteren Kohorten von Studierenden, die ein überwiegend

negatives Bild von Programmierung haben, kann darüber hinaus bei neueren Studierenden eine ablehnende Haltung entstehen.

1.1 Mein gewählter Lehransatz

Mit dem Antritt meiner Professur zum Wintersemester 2020/21 übernahm ich die Lehrveranstaltung »Objektorientierte Programmierung« in den Studiengängen Systems Engineering und Data Science an der Technischen Hochschule Augsburg. Durch modernere pädagogische Ansätze (Savage & Piwek, 2019) und praxisrelevante Softwareprojekte möchte ich die Herausforderungen des Programmierens für Studierende reduzieren. Als Programmiersprache wählte ich Python, da sie einfacher zu erlernen ist als C++ oder Java (Kadar et al., 2021). Die Problemabstraktion, das Zerlegen in Teilprobleme und die Integration der entstehenden Teillösungen in ein resultierendes Gesamtprogramm stehen im Vordergrund der Lehrveranstaltung. Voraussetzungen dafür sind zum einen die Fähigkeit, mehrdimensionale Daten in aufgabengeeignete Strukturen zu bringen, und zum anderen die sinnvolle Erstellung von Klassen zur Kapselung von Programmfunktionalitäten und Daten.[1]

Das Grundkonzept des objektorientierten Programmierens, das Bilden von Klassen, ist am Beispiel der Kategorisierung von Tieren leicht erklärt. Die Basisklasse »Tier« beschreibt generelle Eigenschaften und Vorgänge (Methoden), die alle Tiere gemeinsam besitzen, etwa Variablen für die Größe oder Funktionen/Methoden für die Fortbewegung. Während die Basisklasse »Tier« die Fortbewegung allgemein als Positionsänderung definiert, konkretisieren die abgeleiteten vererbten Klassen »Fisch« und »Vogel« die Fortbewegung spezifischer (Positionsänderung durch Schwimmen bzw. Fliegen). Ein späteres Hinzufügen von Eigenschaften zur Basisklasse »Tier« – z.B. der Energieverbrauch – wirkt sich auf alle abgeleiteten Klassen (»Fisch« und »Vogel«) aus. Dieser Strukturierungsansatz dient zur Vermeidung redundanter, ähnlicher Programmierungen und sorgt für größere Übersichtlichkeit.

Auch wenn das Grundkonzept der Objektorientierung am vorangegangenen Beispiel leicht erklärt werden kann, ist dessen Transfer und Anwendung auf eine andere, praxisbezogene Aufgabenstellung für die Studierenden sehr

1 Als Kapselung wird in der objektorientierten Programmierung der Zugriff auf Methoden und Attribute von Klassen bezeichnet. Das Klassendesign und damit der Umfang der Kapselung setzt aber eine Vorstellung von der sinnvollen hierarchischen Strukturierung des zu entwickelnden Softwareprogramms voraus.

schwierig. Entsprechend den Empfehlungen von Soloway und Spohrer (2013) sowie Savage und Piwek (2019) habe ich mich daher bewusst gegen ein Skript oder Folien entschieden, um den Fokus auf die viel wichtigeren praktischen Programmiererfahrungen zu legen – darunter die Interpretation von Fehlermeldungen sowie die Fehlersuche und -behebung. Darüber hinaus wird der Ratschlag von Edwards (2003) beherzigt, dass die Studierenden ab der ersten Lehrveranstaltung am eigenen Laptop mit installiertem Python Interpreter und Code-Editor programmieren sollten. Somit stehen das Arbeiten am Laptop und das ständige Ausprobieren der eigenen Programmierung im Vordergrund. Über die einzelnen Lehrinhalte hinweg werden während des Semesters zudem zwei Softwareprojekte erstellt, die es den Studierenden erleichtern, die erlernten Programmierkonzepte auf andere Aufgabenstellungen zu übertragen. Anhand dieser Projekte sollen die Vorteile der Aufgabenzerlegung in einzelne Teilprogramme sowie deren Umsetzung in Klassen/Objekten klarer verständlich werden. Von der Eingrenzung bei der Fehlersuche über verschiedene Erweiterungen des Softwareprojekts bis hin zur Modularität werden die Studierenden dabei sukzessive an eine abstraktere Denkweise sowie an die Betrachtung von Teillösungen zur Beherrschung der Komplexität herangeführt.

Als Themen für die semesterbegleitenden Softwareprojekte werden ein Adress- und ein Börsenmanager verwendet. Die Erstellung des Adressmanagers wird schrittweise in der Vorlesung illustriert. Dabei werden nicht nur die verschiedenen Themen und Konzepte der Programmierung aufgegriffen: Anhand der Live-Programmierung können auch unterschiedliche Fehlerszenarien und Programmlaufzeitunterschiede diskutiert und demonstriert werden. Mit dem neuen Wissen aus der Vorlesung stellen die Studierenden sich dann der Transferaufgabe, die gezeigten Lösungsansätze zur sukzessiven Erstellung eines Börsenmanagers zu nutzen. Die Aufgaben werden themenbezogen wöchentlich nach jeder Vorlesung verteilt und durch die Studierenden individuell bearbeitet. Für jede abgegebene Programmierung erhalten die Studierenden Feedback. Die für den Börsenmanager erforderliche Transferleistung ist hoch genug, dass sich jede:r Studierende mit der Thematik und der eigenen Programmierung befassen muss. Gleichzeitig ist die Transferleistung aber immer noch so gering, dass sich die Lernenden auf das Wesentliche konzentrieren können.

Als zusätzlicher Anreiz für die Studierenden werden Punkte für die Abgaben vergeben, die gleichzeitig einen Bonus von einer Notenstufe auf die abschließende Prüfung darstellen. Zur Selbstkontrolle werden zu jeder Aufga-

be Unit-Tests ausgegeben, mit denen die Studierenden ihre Programmierung vor Abgabe kontrollieren können. Unterstützend stehen den Studierenden neben einer Fragerunde in der Vorlesung außerdem ein Tutorium und ein Moodle-Hilfeforum zur Verfügung. Das Tutorium dient dabei nicht nur der Klärung von Fragen, sondern ermöglicht es den Studierenden auch, den jeweiligen Sachverhalt aus einer anderen Perspektive erklärt zu bekommen. Der Vorteil des Peer-Reviews durch die Tutor:innen besteht dabei in der individuellen Betrachtung der programmierten Lösung sowie in der Abwägung der damit einhergehenden Vor- und Nachteile. Darüber hinaus sensibilisiert das Format die Studierenden zusätzlich für die Feinheiten der Programmierung.

1.2 Limitation der konventionellen Lehrevaluation

Anders als erwartet, gelang es mir mit dem eben beschriebenen Lehrkonzept nicht, die Herausforderungen des Programmiereinstiegs für die Studierenden zu reduzieren. Im Sommersemester 2022 kann die Teilnahmeaktivität der 70 Studierenden wie folgt zusammengefasst werden: *Ausbaufähig*.

Bereits ab der dritten von zwölf Transferaufgaben zum Börsenmanager wurden diese nur noch von einem Dittel der Studierenden bearbeitet. Das Hilfeforum enthielt zum Semesterende lediglich fünf Fragen und auch das Tutorium wurde nur sehr spärlich besucht und schrumpfte nach dem ersten Semesterdrittel auf weniger als zehn Studierende. Diese rapide abnehmende Teilnahmeaktivität spiegelte sich auch in der Prüfungsleistung wider – das Histogramm der Notenvergabe entsprach einer Badewannenkurve. Die Studierenden, die sich regelmäßig mit den Transferaufgaben befasst hatten, erzielten zwar Noten besser als 2,0, doch der Rest platzierte sich unterhalb von 3,3 bis »durchgefallen«. Selbst der Anreiz von Bonuspunkten für die Prüfung motivierte viele nicht zur Bearbeitung der Aufgaben. Stattdessen streckten die Bonuspunkte das Tal der Badewannenkurve nur noch weiter, da die zusätzlichen Punkte die Noten derjenigen Studierenden verbesserten, die ohnehin bereits durch die regelmäßige Aufgabenbearbeitung eine gute bis sehr gute Note erhielten.

Kurzum: Irgendetwas lag hier im Argen, doch ich wusste nicht, was.

Zur Identifikation der möglichen didaktischen oder inhaltlichen Unstimmigkeiten in der Lehrveranstaltung, welche die beschriebenen Beobachtungen begünstigen könnten, wurde daher zum Semesterende eine anonyme Lehrevaluation durchgeführt, an der etwa ein Drittel der Studierenden teilnahm. Die dabei gestellten Multiple-Choice-Fragen zur Bewertung der einzelnen

Aspekte der Lehrveranstaltung wurden fast ausschließlich positiv beantwortet. Die Anmerkungen aus den Freitextfeldern für Verbesserungsvorschläge waren entweder kleinere Änderungsempfehlungen oder sehr pauschal formuliert. Zu den Kritikpunkten, die hier angeführt wurden, gehörten u.a. ein zu hoher Zeitaufwand und eine unklare Aufgabenstellung. Konkrete Verbesserungsideen für die Veranstaltung konnten daraus allerdings nicht abgeleitet werden.

Oberflächlich betrachtet, belegt diese Evaluation eine hohe Qualität der Lehrveranstaltung und erfolgreich erreichte Lehrziele. Auf den zweiten, kritischen Blick fallen allerdings die geringe Anzahl an Teilnehmenden und die fast ausschließlich positiven Rückmeldungen auf. Folglich liegt die Vermutung nahe, dass sich überwiegend diejenigen Studierenden an der Umfrage beteiligten, die auch in der Vorlesung präsent waren und regelmäßig die Aufgaben bearbeitet hatten. Daraus würde aber ein verzerrtes Ergebnisbild resultieren, da es bedeuten würde, dass die »unzufriedenen« Studierenden nicht vertreten sind. Selbst bei vereinzelten konkreten Verbesserungsvorschlägen könnten diese durch den Dozierenden aufgrund der mangelnden Signifikanz im Vergleich zu den sonstigen positiven Rückmeldungen vernachlässigt werden. Da die Teilnahme an der Lehrevaluation zum einen anonym und zum anderen freiwillig ist, ist es folglich gar nicht möglich, durch diese konventionelle Form der Lehrevaluation Optimierungspotentiale zu identifizieren.

2. Misfits in der Lehrevaluation

Entsprechend der vorangegangenen Vermutung wird im folgenden Szenario angenommen, dass an einer Lehrevaluation nur Studierende teilnehmen, die mit der Lehrveranstaltung zufrieden oder gar davon begeistert sind. In diesem Szenario spiegeln die Ergebnisse der Lehrevaluation eine verzerrte Perspektive wider, die mit geringer werdender Teilnahmequote verstärkt wird. Infolgedessen werden Dozierende blind für die Verbesserungen, die eigentlich notwendig wären, um auch den weniger begeisterten Studierenden entgegenzukommen. Aufgrund der Anonymität und Freiwilligkeit der Teilnahme – beide Faktoren sind unvermeidlich, da sie für Lehrevaluationen an deutschen Hochschulen Pflicht sind – besteht für Dozierende gar keine andere Möglichkeit, als subjektive Vermutungen über die Verbesserungspotenziale ihrer Lehre anzustellen.

Die Lehrevaluation muss daher in Richtung einer kritischen Reflexion, auch durch die positiv gestimmten Studierenden, gelenkt werden. Entsprechende Frage- bzw. Aufgabenstellungen, etwa »Was würden Sie verbessern?« oder »Nennen Sie drei negative Aspekte der Vorlesung«, sind dabei allerdings nur wenig hilfreich, da sie die Studierenden nicht zum Nachdenken anregen. Alternative Lehrevaluationsansätze wie das Teaching Analysis Poll (TAP) (Johannsen & Meyer, 2023) setzen jedoch die mehrheitliche Teilnahme der Studierenden voraus.

Um mit wenigen, dafür aber durchgehend motivierten Studierenden die Lehrveranstaltung aus Sicht ihrer »unzufriedenen« Kommiliton:innen kritisch zu betrachten und konkrete Verbesserungsvorschläge abzuleiten, müssen die an der Lehrevaluation Teilnehmenden einen Perspektivwechsel vollziehen. Ein solcher Perspektivwechsel kann durch die Misfits von EMPAMOS, der *Empirischen Analyse motivierender Spielelemente* (Voit et al., 2020; Bröker et al., 2021) ermöglicht werden. Die Misfits eignen sich besonders gut für die Lehrevaluation, da sie eine kritische Betrachtung des jeweiligen Sachverhalts erzwingen und durch Abstraktion auf die Spielebene die motivationsverhindernden Aspekte fokussieren. Da EMPAMOS aber vor allem Arbeitsvorgänge und Prozesse im beruflichen sowie wirtschaftlichen Kontext betrachtet, stellt sich die Frage, wie die vorhandenen Misfits in der Lehrevaluation genutzt werden können.

2.1 Gestaltung der Misfit-Lehrevaluation

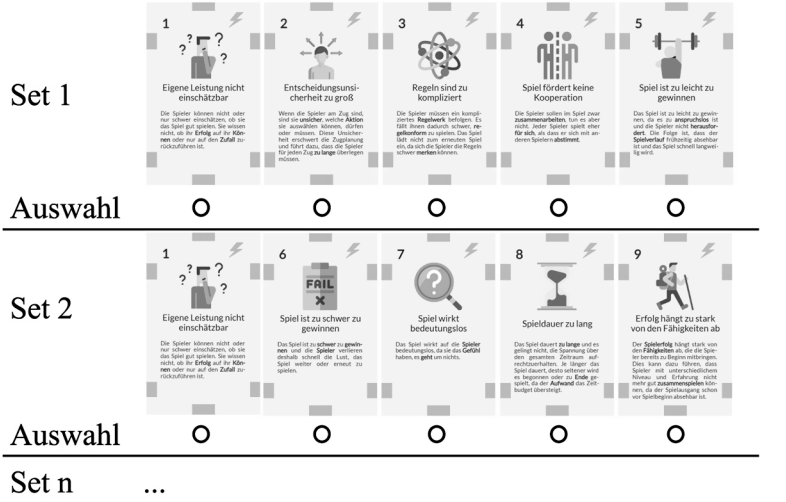
Damit in der Lehrevaluation der Perspektivwechsel unter Zuhilfenahme der Misfit-Karten gelingt, ist es vorab erforderlich, diese Karten kurz vorzustellen. Zur Verdeutlichung kann ein beispielhaftes, allgemein bekanntes Gesellschaftsspiel wie »Mensch ärgere dich nicht« mit Blick auf seine möglichen Misfits – d.h. die demotivierenden Faktoren des Spiels – diskutiert werden. Sehr schnell ergibt sich unter den Studierenden ein gemeinsames Meinungsbild zu verschiedenen Misfits. Durch den anschließenden Vergleich des diskutierten Gemeinschaftsspiels mit der Lehrveranstaltung ist ein ausreichendes Verständnis bei den Studierenden vorhanden, um die Misfits in der Lehrevaluation korrekt zu interpretieren.

Als nächstes stellt sich die Frage, wie die Misfits in der Lehrevaluation eingesetzt werden sollen. Eine Möglichkeit besteht darin, die Studierenden zu bitten, aus den 25 Misfit-Karten des EMPAMOS-Toolkits fünf auszuwählen, die sie für besonders zutreffend halten. Dabei ist sowohl eine gemeinsame

Auswahl mit vorangehender Diskussion unter den Studierenden möglich als auch eine individuelle Auswahl durch die einzelnen Teilnehmenden. Beide Ansätze bergen jedoch das Risiko, dass die Studierenden eine vorurteilsbehaftete Auswahl treffen und nur die offensichtlichsten Misfits – etwa »*Spieldauer zu lange*« – wählen.

Um dieses Risiko zu minimieren, besteht eine andere Möglichkeit darin, die Misfit-Karten in Sets mit einer möglichst hohen Kombinatorik aufzuteilen, wie in Abbildung 1 dargestellt.

Abbildung 1: Beispielhafte Darstellung zweier Misfit-Sets. Im Rahmen der Lehrevaluation erhält jede:r Studierende 25 Sets. Aus einem Set wird stets nur ein Misfit ausgewählt.



Ein Set besteht aus fünf unterschiedlichen Misfit-Karten und die gesamte Lehrevaluation aus insgesamt 25 Sets. Die Studierenden wählen zunächst aus jedem Set die für sie jeweils relevanteste Karte aus. Durch die Bereitstellung unterschiedlich kombinierter Sets sind die Befragten dabei gezwungen, sich mit allen Misfit-Karten auseinanderzusetzen. Dies verhindert zwar nicht unbedingt die Auswahl der offensichtlichen Karten, doch wird der Fokus immerhin auf alle Misfit-Karten gleichmäßig verteilt.

Zusammenfassend ergeben sich für die Umsetzung einer Misfit-unterstützten Lehrevaluation die folgenden fünf Anforderungen in absteigender Priorität;

I. Ein Set, entsprechend einer Frage, besteht aus fünf Misfits.
II. Es werden 25 Sets gebildet; dies entspricht der Anzahl an Fragen in der Evaluation.
III. Die Studierenden dürfen aus einem Set nur ein Misfit auswählen.
IV. Jedes Misfit muss über alle Sets hinweg exakt fünfmal auftreten.
V. Jedes Misfit sollte in Kombination mit einem anderen Misfit nur einmal über alle Sets hinweg erscheinen.

Anforderung V adressiert die maximale Auswahlkombinatorik der Misfits in der Umfrage, um einerseits eine gleichmäßige und vorurteilsfreie Betrachtung aller Misfits zu erzielen. Das folgende Beispiel soll die Anforderung V verdeutlichen.

Set 1 (Abb. 1) besteht aus den Misfits mit den Nummern 1, 2, 3, 4 und 5. Set 2 wird aus den Misfits 1, 6, 7, 8 und 9 gebildet. In allen weiteren Sets soll die Kombination von 1 mit 3, 4, 5, 6, 7, 8 oder 9 ausgeschlossen werden, da diese Kombination in den Sets 1 und 2 bereits vorhanden ist. Vollständig umsetzen lässt sich die Anforderung V nicht, da sonst nur 20 Sets entstehen und die Misfits 22, 23, 24 und 25 nur einmal über alle Sets vorkommen würden. Durch ein eigens geschriebenes Programm wurde die bestmögliche Umsetzung der Anforderung V ermittelt. Die Zusammensetzung der so generierten 25 Misfit-Sets ist in Tabelle 1 dargestellt.

Tabelle 1: Sets für die Misfit-Lehrevaluation. Die Zahlen beziehen sich auf die Nummern der Misfit-Karten im EMPAMOS-Toolkit

Sets, deren Misfits in Kombination zueinander nur 1x auftreten				
6, 18, 17, 19, 23	1, 5, 25, 9, 8	4, 10, 22, 20, 7	13, 14, 2, 21, 3	12, 11, 24, 15, 16
3, 22, 11, 17, 1	2, 7, 15, 23, 25	2, 9, 16, 19, 20	4, 5, 12, 18, 21	6, 8, 10, 14, 24
4, 8, 13, 15, 17	7, 9, 11, 13, 18	1, 10, 16, 21, 23	12, 14, 19, 22, 25	3, 5, 6, 15, 20
5, 7, 14, 16, 17	17, 20, 21, 24, 25	3, 4, 9, 23, 24		

Sets, bei denen eine Misfit-Kombination bereits vorhanden ist				
1, 2, 6, 12, 13	2, 8, 16, 18, 22	5, 10, 11, 13, 19	12, 14, 19, 22, 25	6, 7, 9, 11, 21
3, 8, 10, 1, 24				
Set mit doppelt vorhandener Misfit-Kombination				
20, 18, 4, 15, 23				

Durch die Gestaltung der Sets mit je fünf Misfits entsteht außerdem die Fragestellung, wie die relevanten Misfits von den zufällig ausgewählten zu unterscheiden sind. Basierend auf den Anforderungen I bis III, der Anzahl der Teilnehmenden und der summierten Auswahl jeder Misfit-Karte über alle Antworten aus der Lehrevaluation lässt sich die Auswahlwahrscheinlichkeit eines bestimmten Misfits mittels Bionominalverteilung entsprechend der folgenden Gleichung (1) errechnen:

$$p_{(k,m)} = B(k(m)|p,n) = \binom{n}{k(m)} p^{k(m)}(1\text{-}p)^{n\text{-}k(m)}$$

- m: Misfit-Karte mit der Nummer m
- $k(m)$ Anzahl der gewählten Misfit-Karte m über alle Teilnehmenden hinweg
- $n = 5 \cdot a$: Maximal mögliche Auswahlanzahl einer Misfit-Karte in Abhängigkeit zur Anzahl der Teilnehmenden a
- $p = 0.2$: Wahrscheinlichkeit der Auswahl einer einzelnen Misfit-Karte pro Set

Der Parameter $k(m)$ entspricht der Häufigkeit des gewählten Misfits m, n der Gesamtanzahl der wählbaren Misfit-Karten zusammenhängender Sets und p der Auswahlwahrscheinlichkeit des Misfits m in einem Set. Bei fünf Misfits pro Set beträgt die Auswahlwahrscheinlichkeit eines Misfits $p = 0{,}2$. Bei einer exakten fünfmaligen Verteilung von einem Misfit m über alle Sets hinweg kann eine teilnehmende Person ein Misfit maximal fünfmal wählen (siehe Anforderung IV). Die Gesamtzahl n errechnet sich aus der maximal wählbaren Anzahl eines Misfits (5) multipliziert mit der Anzahl an Teilnehmenden a in der Evaluation: $n = 5 \cdot a$.

Für die Relevanzabschätzung eines bestimmten Misfits werden zwei Hypothesen aufgestellt:

- $H_0\ p = 0{,}2 \rightarrow$ Misfit wurde zufällig gewählt und besitzt damit keine Relevanz.
- $H_1\ p \neq 0{,}2 \rightarrow$ Misfit wurde mehrheitlich gewählt oder ausgeschlossen.

Mit einem gewählten Konfidenzniveau und mittels der inversen kumulativen Verteilungsfunktion werden die Vertrauensintervallgrenzen berechnet. Damit können die eindeutig relevanten, die zufälligen und die irrelevanten Misfits identifiziert werden. Basierend auf den relevanten Misfits können im Weiteren Interpretationen und Schlussfolgerungen bezüglich der Lehrveranstaltung entwickelt werden.

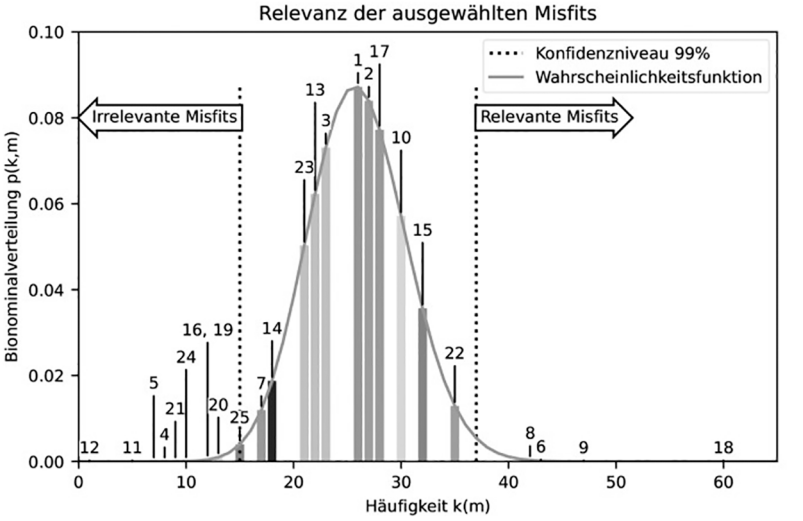
3. Misfit-Lehrevaluation im Modul »Objektorientierte Programmierung« im Sommersemester 2023

Im Rahmen der Lehrveranstaltung »Objektorientierte Programmierung« im Sommersemester 2023 wurde der oben beschriebene Misfit-Evaluationsansatz mit 25 Sets durchgeführt. An der Evaluation nahmen insgesamt 26 Studierende teil. Abbildung 2 veranschaulicht das Verteilungsergebnis der von ihnen gewählten Misfits. Das gewählte Konfidenzintervall kennzeichnet dabei drei Bereiche. Links sind eindeutig irrelevante Misfits, in der Mitte die zufällig gewählten Misfits und rechts eindeutig ausgewählte Misfits.

Mit Blick auf das Diagramm ist allerdings anzumerken, dass es sich dabei nicht um ein Histogramm handelt: Die Abszisse stellt die Auswahlhäufigkeit einer Misfit-Karte dar. So wurde etwa die Misfit-Karte 18 in Summe fast 60-mal durch alle Teilnehmenden ausgewählt. Die Ordinate stellt indes die Wahrscheinlichkeit durch die Bionominalverteilung dar.

Mithilfe der Bionominalverteilung und eines gewählten Konfidenzintervalls können diejenigen Misfits eindeutig identifiziert werden, die alle Befragten als relevant oder irrelevant betrachten (links oder rechts der Konfidenzintervallgrenzen). Alle Misfits innerhalb des Konfidenzintervalls können als zufällig gewählt betrachtet werden. Aus diesem Grund werden für die weitere Auswertung nur die Misfits 18, 9, 6 und 8 verwendet, die als statistisch relevant identifiziert wurden.

Abbildung 2: Ergebnisse der Misfit-Lehrevaluation im Sommersemester 2023



3.1 Persönliche Interpretation der Ergebnisse

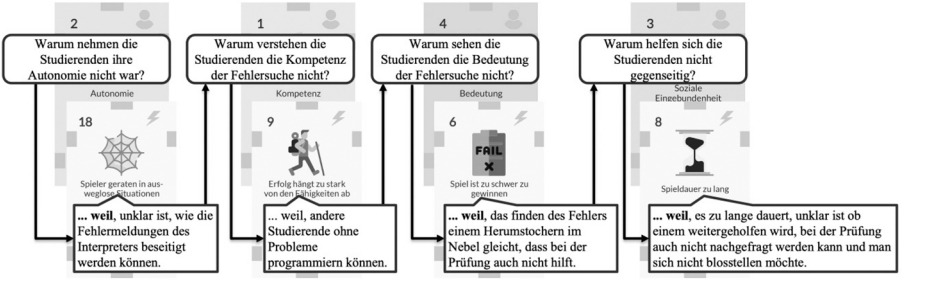
Die folgenden Schlussfolgerungen stellen meine persönliche Interpretation der Ergebnisse der Misfit-Lehrevaluation zum Modul »Objektorientierte Programmierung« im Sommersemester 2023 dar. Für die Interpretation nutze ich die EMPAMOS-Methoden »*Regelkreis-Radar*« und »*Motivationale Auftragsklärung*«. Tabelle 2 fasst die Interpretationsergebnisse durch Anwendung der EMPAMOS-Methode »*Regelkreis-Radar*« zusammen.

Tabelle 2: Interpretation der relevanten Misfits in Anlehnung an die EMPAMOS-Methode »Regelkreis-Radar«

Regelkreis	m	Misfit	Möglicher Grund
Aktion	18	*»Spieler geraten in ausweglose Situationen«*	Viele Studierende verstehen nicht, wie man mit den Fehlermeldungen des Interpreters umgehen soll.
	9	*»Erfolg hängt zu stark von den eigenen Fähigkeiten ab«*	Wer bereits programmieren kann, kann auch die Aufgaben bewältigen.
	8	*»Spieldauer zu lange«*	Die Aufgaben sind zu umfangreich.
Welt	6	*»Spiel ist zu schwer zu gewinnen«*	Für die Studierenden ist lange unklar, warum das Programm nicht läuft.

Entsprechend der Häufigkeit der gewählten Misfits wurden in Anlehnung an die EMPAMOS-Methode »*Motivationale Auftragsklärung*« mögliche Ursachen für das erschwerte Erlernen objektorientierter Programmierung ermittelt. Abbildung 3 veranschaulicht den Erörterungsweg vom jeweiligen Misfit zur interpretierten Ursache.

Abbildung 3: Erörterung der Ursachen für das erschwerte Erlernen der objektorientierten Programmierung



Eine mögliche Schlussfolgerung aus Abbildung 3 könnte in der zusätzlichen Punktevergabe für gute Fragen und Antworten der Studierenden liegen. Dadurch gewinnt das Stellen von Fragen an Bedeutung, denn die dafür ver-

gebenen Punkte münden ebenfalls in Bonuspunkte für die Prüfung. Ein ähnlicher Ansatz wird bereits seit langem in »*Community Question Answering*«-Foren (CQA) – etwa bei *Stackoverflow* – genutzt, um die Forenqualität und die damit verbundene Attraktivität zu erhöhen. Dieses Prinzip könnte auch in Lehrveranstaltungen angewendet werden. Voraussetzung hierfür wäre allerdings, dass das Forum für die Studierenden exklusiv bleibt, denn das sorgt nicht nur für einen geschützten Raum, sondern stellt auch sicher, dass alle Teilnehmenden mit denselben Voraussetzungen starten können.

4. Abschlussbetrachtung

Der Misfit-Evaluationsansatz ermöglicht den teilnehmenden Studierenden einen Perspektivwechsel hin zur kritischen und abstrahierenden Betrachtung der Lehrveranstaltung. Für die Studierenden der Pilot-Evaluation im Modul »Objektorientierte Programmierung« war dieser Ansatz eine spannende und abwechslungsreiche Erfahrung, an der sie sehr motiviert mitarbeiteten. Der im Rahmen der Evaluation vermittelte Einblick in EMPAMOS wurde zudem sehr interessiert aufgenommen. Besonders beeindruckend für die Studierenden war dabei die naheliegende Schlussfolgerung in Form der Bepunktung von Fragen und Antworten, auf die aber niemand im Rahmen einer »konventionellen Evaluation« gekommen wäre. Vor dem Hintergrund der Lehrveranstaltung »Objektorientierte Programmierung« und der mit den Inhalten des Moduls verbundenen Notwendigkeit der Abstraktion stellte die durchgeführte Misfit-Lehrevaluation außerdem ein Paradebeispiel für die Potenziale der Abstraktion dar.

Zeitlich betrachtet ging mit der Konzeption der Misfit-Methode der höchste Aufwand einher. Durch das in diesem Beitrag erläuterte Vorgehen ist die zeitliche Inanspruchnahme durch die Misfit-Lehrevaluation nicht wesentlich höher als bei einer konventionellen Evaluation und liegt für 25 Sets à fünf Misfits bei ungefähr 20 Minuten. Zusätzlich sollten aber noch 20–25 Minuten für die Erläuterung der Misfits und die Diskussion eines einfachen beispielhaften Gesellschaftsspiels eingeplant werden. Die nachgelagerte Erläuterung der Schlussfolgerungen aus der Misfit-Lehrevaluation ist schnell durchgeführt. Bei vielen der Studierenden entsteht jedoch erst im Verlauf der Evaluation die Erkenntnis und auch die Begeisterung für das Vorgehen – zwei Aspekte, die zu den eigentlich interessanten Diskussionen führen, die nicht gestoppt werden sollten. Alles in allem ist es daher ratsam, für die Misfit-

Lehrevaluation an zwei aufeinanderfolgenden Unterrichtstagen jeweils 45 Minuten einzuplanen, um ein möglichst umfassendes Feedback zu erhalten.

Danksagung

Abschließend möchte ich mich bei Prof. Dr. Thomas Voit bedanken, der mir im Rahmen der EMPAMOS-Schulungen 1 und 2 bei der methodischen Umsetzung der Lehrevaluation als Sparringspartner zur Verfügung stand.

Literatur

Bröker, T., Voit, T. & Zinger, B. (2021). Gaming the System: Neue Perspektiven auf das Lernen. In Hochschulforum Digitalisierung (Hg.), *Digitalisierung in Studium und Lehre gemeinsam gestalten. Innovative Formate, Strategien und Netzwerke* (S. 497–513). Springer VS. https://link.springer.com/chapter/10.1007/978-3-658-32849-8_28

Cheah, C. S. (2020). Factors Contributing to the Difficulties in Teaching and Learning of Computer Programming: A Literature Review. *Contemporary Educational Technology*, *12*(2), ep272. https://doi.org/10.30935/cedtech/8247

Dale, N. B. & Weems, C. (2014). *Programming and Problem Solving with C++*. Jones & Bartlett Publishers.

Edwards, S. H. (2003). Rethinking computer science education from a test-first perspective. In *Companion of the 18th annual ACM SIGPLAN conference on object-oriented programming, systems, languages, and applications*. Association for Computing Machinery (Oopsla ›03), 148–155. https://doi.org/10.1145/949344.949390

Johannsen, T. & Meyer, H. (2023). Improving teaching quality in higher education: a practitioner's guide to using formative teaching analysis poll. *European Society for Engineering Education (SEFI)*. https://doi.org/10.21427/8REM-2V61

Ka Kadar, R., Wahab, N. A., Othman, J., Shamsuddin, M. & Mahlan, S. B. (2021). A Study of Difficulties in Teaching and Learning Programming: A Systematic Literature Review. *International Journal of Academic Research in Progressive Education and Development*, *10*(3), 591–605. https://doi.org/10.6007/IJARPED/v10-i3/11100

Konecki, M. (2014). Problems in programming education and means of their improvement. In B. Katalinic (Hg.), *DAAAM international scientific book* (S. 459–470). DAAAM International Vienna.

Lahtinen, E., Ala-Mutka, K. & Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*, *37*(3), 14–18. https://doi.org/10.1145/1151954.1067453.

Savage, S. & Piwek, P. (2019). *Full report on challenges with learning to program and problem solve: an analysis of first year undergraduate Open University distance learning students' online discussions*. The Open University. https://oro.open.ac.uk/68073/

Soloway, E. & Spohrer, J. C. (Hg.) (2013) *Studying the Novice Programmer*. Psychology Press. https://doi.org/10.4324/9781315808321.

Voit, T., Schneider, A. & Kriegbaum, M. (2020). Towards an Empirically Based Gamification Pattern Language using Machine Learning Techniques. In *2020 IEEE 32nd Conference on Software Engineering Education and Training (CSEE&T). 2020 IEEE 32nd International Conference on Software Engineering Education and Training* (S. 1–4). IEEE. https://doi.org/10.1109/CSEET49119.2020.9206223.

Xue, Y., Chen, H., Bai, G. R., Tairas, R. & Huang, Y. (2024). Does ChatGPT Help With Introductory Programming? An Experiment of Students Using ChatGPT in CS1. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training. ICSE-SEET ›24: 46th International Conference on Software Engineering: Software Engineering Education and Training* (S. 331–341). ACM. https://doi.org/10.1145/3639474.3640076

Bildquelle: »Artificial Illustrations« – ein studentisches Projekt des FIDL